A Study on Large Scale Graph Analysis Using Eigen Decomposition and Tensor Decomposition

A Study on Large Scale Graph Analysis Using Eigen Decomposition and Tensor Decomposition

Koji Maruhashi

March 2014

Graduate School of Systems and Information Engineering
University of Tsukuba

Submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy in Engineering

Abstract

The focus of this thesis is anomaly detection and path analysis of large scale, real world graphs. We mainly focus on detecting anomalous sub-structures from large scale *static* and *unlabeled* graph data, whose adjacency information can be represented as matrices or tensors, *i.e.*, *adjacency matrices* or *adjacency tensors.* These graphs include simple undirected graphs, bipartite graphs, and k-partite k-uniform hypergraphs (k,k-hypergraphs).

The main motivation of this thesis is to detect anomalous patterns related to community structures and path capacities in graph data. We consider a clique-like sub-structure as a "community", which is a set of nodes in which almost all possible combinations of nodes have common edges. Path capacity is the number of l-hops paths between two sets of nodes. None of the existing works on anomaly detection in graph data did not consider the path capacities between nodes and community structures.

The core intuition behind this thesis is that nodes can be projected on a low dimensional space in which path capacities between each pair of nodes can be roughly estimated, by low rank approximation of adjacency matrices or adjacency tensors. As we discuss in the preliminary section, the coordinates of nodes have high absolute values for *important* nodes in the graph, and exponentially decay as nodes are separated from the *important* nodes. Most of existing data mining related works using such coordinates pay attention only to high absolute values, and this means that these works only consider the *important* nodes themselves. We utilize the coordinates of even very low absolute values, which enables us to deal with the path capacities between all nodes and the *important* nodes.

Based on above intuition, we tackled three problems in this thesis. (1) Anomaly detection of k,k-hypergraph: We defined anomalous clique-like patterns located far from *important* nodes, and spot such patterns by detecting spikes of coordinates of nodes obtained by tensor decomposition of the adjacency tensor. (2) Pattern discovery of bipartite graph: We spot several connection patterns related to community structures by detecting linear-like patterns from the distributions of the node properties corresponding to singular vectors of the adjacency matrix. (3) Shortest path distance estimation: We estimate the short distances more accurately than existing methods using landmark nodes, by approximating the adjacency matrix multiplied by k using a small number of eigenvalues and eigenvectors of the adjacency matrix.

Acknowledgments

First, I would like to thank my advisors in University of Tsukuba. Hiroyuki Kitagawa, Toshiyuki Amagasa, and Tetsuya Sakurai gave me many helpful advices.

I cannot complete a lot of works in this thesis without great help of Christos Faloutsos and Fan Guo, my collaborators during my stay at Carnegie Mellon University.

The work on this thesis was supported by Fujitsu Laboratories Ltd, and I thank my colleagues in the company, Nobuhiro Yugami, Junichi Shigezumi, and many other colleagues.

I wish to thank my family for supporting my life. I thank my parents and parents-in-law for great help. And, above all, I thank Hiromi, my wife, and Miyuu and Yuji, my children, for always being there and making me happy.

Contents

Chapter 1

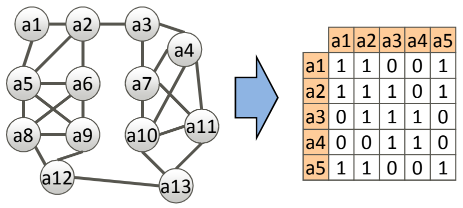
Introduction

1.1 Motivation

Along with development of sensoring techniques, more and more data can be generated and stocked from everywhere in the society. Among them, graph data is ubiquitous: internet, social networks, food web, protein networks and many more. The focus of this thesis is anomaly detection and path analysis of large scale, real world graphs using spectral properties of matrices and tensors corresponding to adjacency information of the graph. We mainly focus on detecting anomalous sub-structure from large scale *static* graph data. This is very important problem in a wide variety of applications such as intrusion detection of computer communication networks, fraud detection in credit card transaction, and so on. In spite of such great importance, we have very little practical solutions. Also, our focus is anomaly detection of *unlabeled*, *undirected* graph data. This is a harder task than that of labeled or directed graph data, because we have less information to distinguish anomalous sub-structures from normal sub-structures.

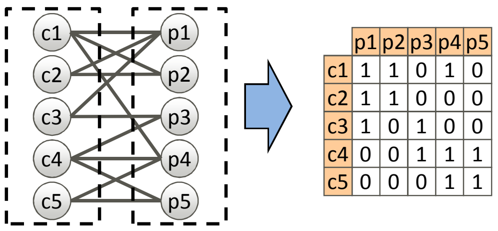
This thesis focuses on graphs whose adjacency information can be represented as matrices or tensors, *i.e.*, *adjacency matrices* or *adjacency tensors*. A simple undirected graph, *e.g.*, friendship relations of social networks based on e-mail communication of *who-sent-whom*, can be represented as a symmetric adjacency matrix (Figure 1.1 (a)). A bipartite graph is a graph whose nodes are divided into two disjoint partitions. For example, purchasing behavior of *who-bought-what* or communication between client IP address and server IP address are bipartite graphs in which customers or client IPs are nodes of a partition and products or server IPs are of another partition. A bipartite graph can be represented as an unsymmetric adjacency matrix

(Figure 1.1 (b)). Moreover, a collection of relations of more than two attributes is regarded as a k-partite k-uniform hypergraph (k,k-hypergraph). A hypergraph is a graph in which each edge have two or more than two nodes, and a k,k-hypergraph is a hypergraph in which nodes are divided into k partitions (k-partite) and all edges have just k nodes (k-uniform) of different partitions. For example, social bookmarks are collections of relations of users and contents and tags, and network traffic logs are those of source IPs and destination IPs and source ports and destination ports. A k,k-hypergraph can be represented as an adjacency tensor, *i.e.*, a multidimensional array (Figure 1.1 (c)).



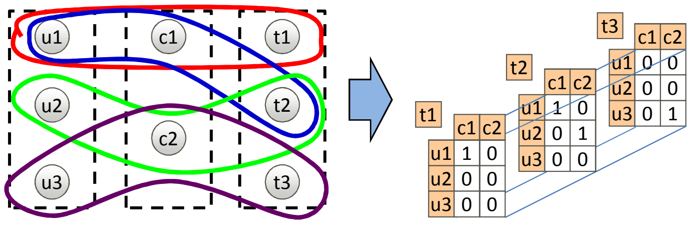
	a1	a2	a3	a4	a5
a1	1	1	0	0	1
a2	1	1	1	0	1
a3	0	1	1	1	0
a4	0	0	1	1	0
a5	1	1	0	0	1

(a) simple graph and a symmetric adjacency matrix



	p1	p2	p3	p4	p5
c1	1	1	0	1	0
c2	1	1	0	0	0
c3	1	0	1	0	0
c4	0	0	1	1	1
c5	0	0	0	1	1

(b) bipartite graph and an adjacency matrix



t1	c1	c2
u1	1	0
u2	0	0
u3	0	0

t2	c1	c2
u1	1	0
u2	0	1
u3	0	0

t3	c1	c2
u1	0	0
u2	0	0
u3	0	1

(c) k,k-hypergraph and an adjacency tensor

Figure 1.1: Graphs and related adjacency matrix and tensor.

Researchers have cast several data mining tasks related to graph data including community detection [122, 121], link prediction [105], centrality analysis [19, 20], and so on. However, anomaly detection of graph data has only recently gained attention. Some existing works on anomaly detection of graph data detected anomalous sub-graphs using variants of the Minimum

Description Length (MDL) principle[126, 50, 30]. Another works applied matrix factorization to matrices corresponding to adjacency information of graph data and looked at the reconstruction error, that is, certain norms of the residual matrices [150, 157, 156]. Recently, a powerful approach was proposed which found *laws* of features of ego-net, *i.e.*, sub-graphs of neighboring nodes of each node, and detected abnormal nodes that deviated from the *laws* [7]. Some works detected anomalous nodes which did not similar to most of other nodes within the same community [58], or nearest community [66].

The main motivation of this thesis is to detect anomalous patterns related to community structures and path capacities in graph data, which have not been taken into account by any of the existing works. A community structure is a group of nodes in which each set of nodes is densely connected internally. In other words, pairs of nodes are more likely to be connected if both of them are members of the same community, and less likely to be connected if they do not share communities. For example, in purchasing data, community structures are groups of products bought by many common customers, and groups of customers who bought many common products. Several methods for community detection have been developed, including minimum-cut method [164], modularity maximization [122, 121], and so on. In this thesis we consider a "community" as a clique-like substructure, that is, a set of nodes in which almost all possible combinations of nodes have common edges. Path capacity [68] is the number of l-hops paths between two sets of nodes in a simple graph or a bipartite graph. For a k,k-hypergraph, we define a sequence of l hyperedges as a l-hops path between a node pair, in which each end hyperedge contains each node of the node pair and every successive hyperedges share at least one common nodes. If two sets of nodes have many l-hops paths of small l, they are well connected. In other words, if they do not have any paths of small l at all, the two sets of nodes are hardly related to each other. We regard well-connected nodes in a graph as *important* nodes, *i.e.*, nodes which have many l-hops paths of small l to many nodes in the graph.

One of the questions we consider is how to realize that there is an abnormally large community whose nodes are hardly related to the *important* nodes which are well related to many nodes in the graph. In social bookmarks, if there exist groups of a lot of user accounts who bookmarked strangely same contents with almost same tags, and these contents are hardly bookmarked by other users, *i.e.*, hardly related to the *important* contents, they are suspected to be spam activities to attract much attention illegally. This is not a trivial question because we must extract anomalous communities in both views of the size of communities and the path capacities between the communities and the *important* nodes.

Another question is how to spot a node or a group of nodes whose connection pattern with several communities is not *normal*, *i.e.*, a node or a group of nodes whose path capacity of one-hop paths between several communities is not *normal*. For example, in a purchasing data, if a group of products which are usually bought by specific customer segments are bought by some customers who belong to other segments, *e.g.*, cosmetics usually bought by groups of young women are bought by some men in strangely much amount, they are suspicious. In a computer communication network, if a set of servers which are usually accessed in a distributed manner, *e.g.*, a group of mirror web-servers, are accessed at strangely much amount on all of the servers (in *not* a distributed manner), it can be declared to be anomalous access. However, the *normal* connection patterns among a potentially vast amount of community structures in real world data include too many possible types to deal with them comprehensively. As a result, it is a hard task to spot the *anomalous* connection patterns among community structures effectively.

The core intuition behind this thesis is that nodes can be projected on a low dimensional space by low rank approximation of adjacency matrices or adjacency tensors. By using the coordinates of node in the low dimensional space, we can roughly estimate path capacities between each pair of nodes, and also detect anomalies in both views of path capacities and community structures, in which most of pairs of nodes have many one-hop paths. As we discuss in Section 3.2, the coordinates of nodes in the low dimensional space obtained by low rank approximation of adjacency matrices or tensors have high absolute values for *important* nodes in the graph, and exponentially decay as nodes are separated from the *important* nodes. The coordinates as values of entries of the 1st singular vectors of an adjacency matrix are called Hub and Authority scores in the famous ranking algorithm HITS proposed by Kleinberg [90]. Most of existing data mining related works using such coordinates pay attention only to high absolute values, and this means that these works only consider the *important* nodes themselves. We utilize the coordinates of even very low absolute values in order to project whole graph structure on low dimensional space, which enables us to deal with the path capacities between all nodes and the communities including the *important* nodes. This is the remarkable point of this thesis.

Moreover, we propose a novel method to estimate the shortest path distance between nodes using such properties of the coordinates of nodes. As the coordinates exponentially decay according to the distance from the *important* nodes, we should be able to estimate the distance between two nodes by considering how many times the coordinates decay. This intuition can be translated into the estimation of path capacity between two nodes, *i.e.*, the number of the shortest paths between two nodes, by approximating the adjacency matrix multiplied by k using a small number of eigenvalues and eigenvectors of the adjacency matrix.

1.2 Overview of the Thesis

Following the motivation described above, we organize this thesis as three parts.

Anomaly detection of k,k-hypergraph (Chapter 4): Suspicious activities such as spam bookmarks and distributed cyber attacks tend to form clique-like patterns of a k,k-hypergraph, in which almost every combinations of k nodes straddling over k partitions share common hyperedges. Moreover, their nodes are hardly related to the *important* nodes, *i.e.*, there are few l-hops paths of small l to the *important* nodes which have many l-hops paths of small l to many nodes in the graph. We spot such clique-like anomalous patterns by detecting spikes of coordinates of nodes obtained by tensor decomposition of the adjacency tensor. We use CP decomposition, one of the well-known tensor decomposition [97]. We show that our method can detect synthesized anomalies effectively, and the computation time scale linearly at the size of hyperedges. Also, we introduce some visualization and summarization of detected spikes for further inspection, and we empirically show some examples of anomalies detected in real-world datasets from three dissimilar application domains. This work was published as [110] and [111].

Pattern discovery of bipartite graph (Chapter 5): To spot a node or a group of nodes whose connection pattern with several communities is not *normal*, we want to know whether there exist any *normal* patterns, *i.e.*, any statistical models about (one-hop) paths between each node and several communities. This is analogous to the *laws* of the connection patterns between each node and “ego-net” analyzed in [7]. However, the *normal* connection patterns among a potentially vast amount of community structures in real world data include too many possible types to deal with them comprehensively. Our proposed method projects all nodes into several 2-D plots corresponding to several singular vectors of the adjacency matrix, and detect remarkable patterns such as linear-like patterns by using Hough transform. We can spot possible *normal* patterns of the nodes related to community structures, *e.g.*, a pattern of the ratio of the degrees and the number of paths to a community, a pattern of the ratio of connections straddling two communities, and so on. Our intuition is that the value of a singular vector corresponding to a node is related to the path capacity, *i.e.*, the number of one-hop paths, between the node and a specific community structure, and that the values of different singular vectors are related to different *important* nodes. We empirically show our method can spot some interesting *patterns* in real-world datasets with millions of edges. This work was published as [109].

Shortest path distance estimation (Chapter 6): If the (i,j)-th element of the adjacency matrix multiplied by $k-1$ is 0 and that multiplied by k is more than 0, the shortest path distance between the i-th and j-th nodes is k. We approximate this procedure by using a small number of eigenvalues and eigenvectors of the adjacency matrix. Our method can estimate the short distances more accurately than existing methods using landmark nodes [134, 125, 171] with almost the same computation cost. Also, we show that there is instability in the accuracy depending on the number of eigenvalues used, *i.e.*, the accuracy of approximation of values of an adjacency matrix varies according to the choice of number of eigenvalues, and it even become worse when we use many eigenvalues. We introduce a heuristic to tackle this problem, and show the heuristic can suppress such instability. This work was published as [112].

Prior to the detailed explanation of each work, we introduce more related works in Chapter 2. We describe some definitions and the common properties of low rank approximation of adjacency matrices or adjacency tensors, along with a summary of datasets we used in this thesis in Chapter 3.

Chapter 2

Related Works

A problem of anomaly detection has been researched within diverse research areas and application domains. Many existing works related to anomaly detection can be categorized by techniques used, and we can categorize the works on anomaly detection of graph data according to the same categories as described in Section 2.1. However, none of the existing works did not consider the path capacities between nodes and communities, even though the clustering-based anomaly detection tried to detect anomalies related to communities.

Community structures can be detected in many ways as described in Section 2.2. Also, several kinds of centralities have been used to measure the *importance* of nodes in a graph data as described in Section 2.3. Among them, eigenvectors of an adjacency matrix can be used as both of a technique of community detection and a centrality measure, *i.e.*, *eigenvector centrality.* This is related to a property of eigenvectors, *i.e.*, the values of the elements of the eigenvectors exponentially decay according to the distances from the *important* nodes, as we will decribe in Section 3.2. Vectors obtained by singular value decomposition and tensor decomposition have similar properties as eigen decomposition (also discussed in Section 3.2). We use these vectors for anomaly detection considering the path capacities between all nodes and communities including the *important* nodes, neither only for community detection nor only for measuring importance of nodes.

Moreover, shortest path distances between pairs of nodes can be estimated by using the landmark nodes selected based on several centrality measures as described in Section 2.3. However, these landmark-based methods tend to err in short distances because a small number of landmark nodes cannot cover many shortest paths of short distances. We tackle this problem by using an adjacency matrix approximated by a small number of eigenvalues and eigenvectors, which have approximated information of the path capacities among nodes in the graph.

Tensor decomposition is a basic technique we use in Chapter 4. Several kind of tensor decompositions with different objective functions have been developed as described in Section 2.4. We use *CANDECOMP/PARAFAC (CP) Decomposition* for anomaly detection of k,k-hypergraph, because it has similar properties to low rank approximation of matrices, *i.e.*, eigen decomposition and singular value decomposition as we will discuss in Section 3.2.

2.1 Anomaly Detection

Anomaly detection is related to the problem of finding patterns in data caused by unexpected behavior. Such patterns are often referred to as anomalies, outliers, exceptions, peculiarities, surprises, and so on. Of these, anomalies and outliers are two terms used most commonly in the context of anomaly detection, and they are sometimes interchangeably. Anomalies are translated into significant real life entities such as cyber intrusions and credit card frauds.

2.1.1 Types of Anomaly Detection

As the definition of the anomalies is hard to give, several different defitions have been given [72, 14, 82, 94]. Chandola [32] gave a comprehensive survey on anomaly detection techniques, and categorized the anomaly detection techniques into *classification based*, *nearest-neighbor based*, *clustering based*, *statistical*, *information theoretic*, and *spectral* techniques.

Classification based techniques learn a classifier using the available labeled training data, and classifies a test instance as normal or anomalous using the classifier. A well-known classification based anomaly detection technique is one-class SVMs [140], which learn a discriminative boundary around the normal instances, and any test instance that does not fall within the learnt boundary is declared as anomalous.

Nearest-neighbor based techniques define an n-dimensional point as an outlier if it is too far away from the rest, and thus lives in a low-density area. These methods not only flag a point as an outlier but they also give outlierness scores. LOF [23] is a typical method in which the LOF score of a data instance is equal to ratio of average *local density* of the k nearest neighbors of the instance and that of the data instance itself. For a normal instance lying in a dense region, its local density will be similar to that of its neighbors, while for an anomalous instance, its local density will be lower than that of its nearest neighbors. Hence the anomalous instance will get a higher LOF score. LOCI [131] finds not only anomalous instances but also anomalous micro-clusters, by using the inverse of the standard deviation of the local densities of the nearest neighbors of the given data instance. LSOD [165] is a scalable alternative leveraged by Locality Sensitive Hashing (LSH) [59]. Many other density-based [5, 9, 34, 65, 88, 124, 170, 101, 81] or distance-based [94, 93, 95] methods that perform well in detecting outliers in very large datasets of high dimension were proposed. Hayashi et al. [73] projected all data points onto a low-dimensional space by conducting tensor decomposition techniques, and detected anomalous points by applying LOF [23].

Clustering based techniques reveal outliers as a by-product of clustering algorithms. One approach is to apply a known clustering algorithm such as DBSCAN [52], ROCK [64], and SPARCL [33] to the data set and declare any data instance that does not belong to any cluster as anomalous. Another approach is to cluster the data by using a clustering algorithm such as SOM [96], and to calculate the distance from each data instance to its closest cluster centroid as the anomaly score of the instance [146].Yet another approach is to declare instances belonging to clusters whose size and/or density is below a threshold as anomalous. The CBLOF score [74] captures the size of the cluster to which the data instance belongs, as well as the distance of the data instance to its cluster centroid.

Statistical techniques fit a statistical model to the given data, and unseen instances are declared as anomalies if they have a low probability to be generated from the model. Statistical techniques form two classes, *non-parametric* and *parametric*, and both techniques have been applied to fit a statistical model. Non-parametric techniques do not generally assume any underlying distribution [45], whereas parametric techniques assume the underlying distribution and estimate the parameters from the given data [51]. Several variations of non-parametric techniques were proposed that captured the conditional dependencies between the different attributes using more complex Bayesian networks, especially for data with categorical attributes [42, 168, 25]. Also, Das et al. [43] proposed an anomaly pattern detection in noisy categorical datasets based on a rule-based anomaly detection [167], by using a measure similar to the concept of *suspicious coincidence* proposed by Barlow [13]. As a parametric technique, Maruhashi *et al.* modeled the distribution of the number of records in count data as mixture of several Poisson distributions by using *Poisson Tensor Factorization* [35], and spotted anomalous behaviors by conducting additional tensor decomposition [113].

Information theoretic techniques analyze the information content of a data set using different information theoretic measures such as *Kolomogorov Complexity*, *entropy*, *relative entropy*, etc. All instances which induce irregularities in the information content of the data set are deemed as anomalous. Arning *et al.* [9] measured the Kolomogorov Complexity of data represented as a string by using the size of the regular expression. Keogh *et al.* [89] used the size of the compressed data file which was compressed by any standard compression algorithm, as a measure of the data set's Kolomogorov Complexity. Recently, Smets and Vreeken [145] took a pattern-based compression approach and employed KRIMP [144] as its compressor for anomaly detection.

Spectral techniques try to find subspaces (embeddings, projections, etc.) by using techniques like Principal Component Analysis (PCA) in which the anomalous instances can be easily identified. A problem of PCA is that outliers strongly affect the principal components which model the correlation structure of *normal* instances. Robust PCA [26] can handle such a problem and assign anomaly scores based on the points' distances from the principal components [143, 57].

2.1.2 Anomaly Detection of Graph Data

Anomaly detection of graph data has only recently gained attention. Here we introduce works on anomaly detection of graph data according to the categories described above.

Several information theoretic techniques were developed for anomaly detection of graph data. Noble and Cook [126] detected repetitive subgraphs within large graphs using *Subdue* system [76] which evaluated subgraphs based on the Minimum Description Length (MDL) principle, and flagged less frequent and not too small / large subgraphs as anomalous. Eberle and Holder [50] also used the MDL principle as well as other probabilistic measures to detect several types of anomalies such as unexpected or missing nodes and edges. Chakrabarti [30] reordered the adjacency matrix so that the compression costs were minimized, *i.e.*, similar nodes were clustered with each other, and spotted outlier edges that lay across clusters.

Spectral techniques utilize low-rank approximation of the adjacency matrices of the graphs. Some methods detect anomalous structures based on the reconstruction error of matrix factorization. These methods use certain norms of the residual matrix of several kinds of matrix factorizations such as CMD [150] and *Colibri* [157]. Tong *et al.* used non-negative matrix factorization which non-negative constraint were added to both low-rank matrix and residual matrix [156]. Ide and Kashima [78] tackled a problem of anomaly detection in a time series of graphs by using a two-step spectral technique. They considered the principal components of the adjacncy matrices at every time instances as the *activity vectors*, and regarded the principal left singular vector of the matrix whose column vectors were the activity vectors within a specific time slice as the normal dependencies over time. The angle between the vector of normal dependencies and a new activity vector was used as the anomaly score of a new graph.

Recently a powerful statistical technique has been proposed. Akoglu *et al.* found the *laws* of the connection patterns between each node and "ego-net", *i.e.*, sub-graphs of neighboring nodes of each node, and detected abnormal nodes that deviated from the *laws* [7].

Moreover, as a nearest-neighbor based technique, Sun *et al.* assessed the normality of nodes in bipartite graphs by using proximity and random walks [148].

Clustering based techniques detect outliers in view of community structures. Gao *et al.* spotted anomalous nodes that did not well belong to the community they resided, which they called community outliers [58]. Gupta *et al.* [66] proposed a concept of *Community Distribution Outliers* in a *heterogeneous* network, *i.e.*, a graph with several types of nodes. They found outlier nodes whose distributions of labels of connected nodes differed very much from those of the nodes within closest community detected by NetClus algorithm [151]. However, even such clustering based techniques do not consider the path capacities between all nodes and communities, *i.e.*, they do not focus on anomalous communities separated by several hops from the *important* nodes, or they do not determine *normal* connection patterns between nodes and communities.

2.2 Community Detection

The basic notion of a community structure is a group of nodes in which each set of nodes is densely connected internally. In other word, pairs of nodes are more likely to be connected if both of them are members of the same community, and less likely to be connected if they do not share communities. However, there are numerous alternative methods for detecting community structures, and there are several varieties of community structures according to the objective function of each method [139, 53].

2.2.1 Similarity-based Method

Similarity measures between nodes in a graph data have been traditionally used to find communities. Commonly used measures include the cosine similarity, the Jaccard index, and the Hamming distance between rows of the adjacency matrix. Hierarchical clustering reveals the multilevel structure of the graph by identifying groups of nodes with high similarity. Hierarchical clustering is very common in social network analysis, biology, engineering, marketing, etc [138, 77].

Some methods embed the nodes in a metric space according to the dissimilarity measures, and separate the nodes in k clusters by using partitioning methods such as k-means and k-medians [75]. Rattingan *et al* [137] adopted k-medoids algorithm and the *Girvan-Newman* method [60] based on edge betweenness centrality.

2.2.2 Cut-based Method

One of the most widely used community detection methods is spectral clustering, a "cut-based" method for understanding graph structures. It partitions nodes based on the eigenvectors of unnormalized Laplacian matrix [116, 117] or normalized Laplacian matrix [36]. The unnormalized Laplacian matrix is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is a diagonal matrix called *degree matrix* whose diagonal elements are degrees of every nodes, and $\mathbf{A}$ is the adjacency matrix. There are several kind of normalized Laplacian matrix such as $\mathbf{L_{sym}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ and $\mathbf{L_{rw}} = \mathbf{D}^{-1}\mathbf{L}$. In short, these methods compute the first k eigenvectors of Laplacian matrix, and divide nodes by some clustering algorithm like k-means according to row vectors of the matrix containing the k eigenvectors as columns. There are several different versions of spectral clustering, depending which kind of the Laplacian matrix is used. A comprehensive survey of spectral clustering was given in [164]. These methods were successfully applied in areas like machine-learning [123] and image segmentation[142].

An alternate approach to partitioning graphs based on cut-based metrics utilizes multilevel algorithms like Metis [86], which coarsen the graph by coalescing nodes and then apply refinement steps to recover partitions. Graclus [46] is also a multilevel algorithm but removes the restriction of equal-sized clusters by using kernel k-means to optimize weighted graph cuts.

2.2.3 Information-theoretic Method

Another partitioning methodology that has been recently proposed takes an information-theoretic view of clustering wherein the graph connectivity matrix is viewed as a set of codewords. Co-clustering [47] divides rows and columns of the matrix into disjoint groups by seeking to maximize mutual information of row groups and column groups subject to constraints on the number of row and column clusters. It requires *a priori* information on the number of clusters like k-means. Cross-Association [31] also creates disjoint row groups and column groups, but it automatically determines parameters such as number of row groups and column groups based on Minimum Description Length (MDL) principle. A simple graph or a bipartite graph can be partitioned by applying these methods to the adjacency matrix of the graph.

2.2.4 Modularity-based Method

Community detection methods based on maximizing the *modularity* metric [122] compare the graph's community structure against a random graph. In short, the *expected* number of edges between nodes i and j if edges are placed at random is $k_i k_j/2m$, where k_i and k_j are the degrees of the nodes and $m = \frac{1}{2}\sum_i k_i$ is the total number of edges in the graph. The *modularity* metric can be written $Q = \frac{1}{4m}\mathbf{s}^T\mathbf{B}\mathbf{s}$, where $s_i = 1$ if node i belongs to a specific group and $s_i = -1$ otherwise, and $\mathbf{B}$ is the *modularity matrix* defined as $b_{ij} = a_{ij} - \frac{k_i k_j}{2m}$, a_{ij} is an element of the adjacency matrix $\mathbf{A}$. Some modularity-based community detection were proposed including a greedy search heuristic[37], and utilizing the spectral properties of a modularity matrix as an alternative to Laplacian matrices [121, 166].

Also, modularity-based community detection was generalized to study community structure in hypergraph. Neubauer *et al.* addressed problems of modularity-based community detection in k,k-hypergraph in view of *connectivity*, *community structure*, and *hyper-incidence*, and discussed how to overcome these problems [120]. Liu *et al.* defined community structures in a k,k-hypergraph as disjoint communities in each partitions such that nodes in the same community were "parallel" and hyperedges between communities were either dense or sparse, and proposed an algorithm for optimizing the quality function [104]. Lin *et al.* modeled multi-relational and multi-dimensional social data as a hypergraph representation and tensor factorization method was applied to extract community structure from the representation [103].

2.2.5 Clique-based Method

A clique is a subgraph in which every node is connected to every other node in the clique, and a node can be a member of more than one clique. One approach of clique-based community detection is to find the maximal cliques, which are not the subgraph of any other clique [107]. Pseudo cliques are natural extension of cliques which are subgraphs obtained by removing small number of edges from cliques, and an algorithm to enumerate all pseudo cliques in a given graph was proposed [162]. A quasi-cliques is an alternative definition of a pseudo clique, and an efficient algorithm was developed [133].

The clique percolation method [128] defines a k-clique community as a union of k-cliques (cliques with k nodes) that can be reached from each other through a series of adjacent k-cliques, where adjacency means sharing $k-1$ nodes, and extracts k-clique communities by locating the maximal cliques rather than the individual k-cliques.

Moreover, near cliques or near-bipartite cores are related to high absolute values of elements of the first few singular vectors of an adjacency matrix [135, 84]. Eigenvectors of matrices are also used in the spectral clustering of the cut-based methods, but the eigenvectors or singular vectors of an adjacency matrix indicate the path capacity of the graph [68, 158], whereas the eigenvectors of a Laplacian matrix indicate the connectivity of the graph [36]. Our method proposed in Chapter 5 also utilize the singular vectors of an adjacency matrix. However, we utilize even very low absolute values to analyze the path capacities between all nodes and communities, whereas existing works only utilize very high absolute values. Other methods proposed in Chapter 4 and Chapter 6 also utilize very low absolute values of the vectors obtained by conducting low rank approximation of adjacency matrices / tensors, whereas other existing works only utilize very high absolute values.

2.3 Path Analysis

Relative importance of a node has been traditionally measured by the concept of *centrality* based on path capacities. Eigenvectors of an adjacency matrix can also be used as a centrality measure, *i.e.*, eigenvector centrality. Nodes with high eigenvector centrality are *important* node.

Also, we describe some algorithms of shortest path distance estimation, and show how these algorithms select *landmark* nodes based on several centrality measures, and our method proposed in Chapter 6 tackles a problem which these landmark-based algorithms have.

2.3.1 Centrality Analysis

Centrality is a fundamental concept in network analysis, which measures relative importance of a node within a graph [20]. Freeman [55] reviewed a number of published measures and reduced them to three basic concepts, *degree centrality*, *closeness centrality*, and *betweenness centrality*.

Degree centrality is a count of the number of edges incident upon a given node. Mathematically, degree centrality of a node i is $c_i = \sum_{j=1}^{I} a_{ij}$ where I is number of all nodes and a_{ij} is the value of the (i, j)-th element of the adjacency matrix $\mathbf{A}$. *Katz centrality* [87] is a generalization of degree centrality, which is a weighted count of the number of paths originating (or terminating) at a given node. In short, Katz centrality of a node i is $c_i = \sum_{k=1}^{\infty} \sum_{j=1}^{I} \alpha^k a_{ij}^{(k)}$, where $a_{ij}^{(k)}$ is the value of the (i, j)-th element of the adjacency matrix multiplied by k, *i.e.*, $\mathbf{A}^k$, and α is an attenuation factor between 0 and 1.

Closeness centrality is classically defined as the total distance from a given node to all other nodes. Mathematically, classical closeness centrality of a node i is $c_i = \sum_{j=1}^{I} d_{ij}$ where d_{ij} is shortest path distance from node i to node j. Since its larger values indicate less centrality, it technically measures farness rather than closeness. Other variants of closeness centrality can be obtained by varying the way the initial distance matrix is defined, *e.g.*, the speed with which randomly walking messages reach a node from elsewhere in the network [127, 147].

Betweenness centrality of a node is the number of times that any actor needs a given actor to reach any other actor [56]. Mathematically, betweenness centrality of a node i is $c_i = \sum_{j=1}^{I} \sum_{k=1}^{I} \frac{g_{jik}}{g_{jk}}$ where g_{jk} denote the number of paths from node j to node k, and let g_{jik} denote the number of paths from j to k that pass through intermediary i. Unfortunately, the computation cost of calculating betweenness centrality is very expensive. The fastest known algorithms to compute betweenness centrality exactly are described by Brandes [22]. Bader *et al.* [12] discuss how to approximate betweenness centrality by random sampling.

Moreover, *eigenvector centrality*, also called *Bonacich's centrality*, is a measure of the influence of a node [18]. Eigenvector centrality can be calculated as the eigenvector of a matrix with the largest eigenvalue, where the matrix is related to adjacency information of a graph. Variants of eigenvector centrality are used in ranking algorithm of web pages such as HITS [90] and PageRank [24].

Eigenvector centrality assigns relative scores to all nodes based on the concept that connections to high-scoring nodes contribute more to the score than equal connections to low-scoring nodes. This concept can be generalized for other low rank approximation of adjacency matrices / tensors, *i.e.*, singular value decomposition and tensor decomposition, as we will discuss in Section 3.2. We use the values of the vectors obtained by low rank approximation of matrices / tensors not only to measure the *importance* of the nodes but also to analyze path capacities between all nodes and the *important* nodes, by utilizing even very low absolute values.

2.3.2 Shortest Path Distance Estimation

Dijkstra's algorithm [48] is the most well-known algorithm for computing shortest paths in weighted graphs. A* algorithm [70] uses a Best-First-Search which favors nodes that are close to the goal, combining the Dijkstra's algorithm which favors nodes that are close to the starting point. ALT algorithms are state-of-the-art in point to point shortest path queries, which prune the search space of the A* algorithm by calculating lower-bounds based on landmark nodes and the triangle inequality [61, 79].

There are algorithms which estimate *approximated* shortest path distance faster than such exact shortest path algorithms by several orders of magnitude [134]. Most of these algorithms first compute coordinates for a few selected nodes as *landmark* nodes, and use the coordinates of these nodes as fixed points to calibrate coordinate values for a given node pair [125, 152, 171, 172, 108]. Global Network Positioning (GNP) [125] is a state-of-the-art which models a network as a geometric space and characterizes the position of any node. GNP first computes positions of the landmark nodes, and then computes positions of all other nodes so as to minimize the error of estimated distances from the landmark nodes by using the Simplex Downhill method [119]. However, the cost of computing positions of GNP is very expensive, and Orion bootstrap [171] tackles this problem by dividing the landmark nodes into two groups. The problem of approximating network distances in real networks by embedding a small set of landmarks has been discussed by Kleinberg et al. [91], and the metric for embedding landmarks was generalized by Abraham et al. [3]. Potamias et al. [134] proved the problem of optimal landmark selection in a graph is NP-hard. They evaluated the accuracy of landmark-based methods with upper bounds and lower bounds of the distances estimated by using the landmark nodes. They compared three kinds of landmark selection strategies, random sampling, highest degree centrality, and highest closeness centrality. And they showed that in practice, simple intuitive landmark selection strategies such as degree centrality and closeness centrality work much better than random ones.

Moreover, the decentralized methods are similar to landmark-based method but compute coordinates of another space for a node pair by using any nodes of which coordinates have already been computed [39, 41]. The main problem with decentralized methods is that the convergence time increases as the number of already computed nodes grows.

However, landmark-based methods tend to err in approximating short distances, since it is difficult for a few landmarks to cover the many shortest paths of close node pairs in a large graph. We tackle this problem by using an adjacency matrix approximated by a small number of eigenvalues and eigenvectors in Chapter 6. Our method is closely related to the work of Tsourakakis [160]. Tsourakakis estimated the number of 3-hop paths returning to the starting nodes from the diagonal elements of a 3 times multiplied adjacency matrix approximated by a small number of eigenvalues and eigenvectors. We estimate the number of k-hops paths for every k as the similar way, and guess the shortest path distances between pairs of nodes based on the approximated information of the path capacities.

2.4 Tensor Decomposition

Tensor decomposition techniques have recently attract much attention in the data mining research community. Several kinds of tensor decompositions with different objective functions have been developed. *CANDECOMP/PARAFAC (CP) Decomposition* and *Tucker Decomposition* are two well-known approaches. Non-negative tensor factorizations retain the non-negative characteristics of the original data, and Boolean Tensor Factorizations factorize a binary tensor to binary factors using Boolean arithmetic. We use *CANDECOMP/PARAFAC (CP) Decomposition* for anomaly detection of k,k-hypergraph in Chapter 4, because it has similar properties to low rank approximation of matrices, *i.e.*, eigen decomposition and singular value decomposition as we will discuss in Section 3.2.

We overview the several kind of tensor decomposition techniques in this Section.

2.4.1 Tucker and CP Decomposition

Tensor decomposition is a basic technique that has been well studied and applied to a wide range of disciplines and scenarios. *CANDECOMP/PARAFAC (CP) Decomposition* and *Tucker Decomposition* are two well-known approaches, and an informative survey on tensor decompositions is presented by Kolda and Bader with many further references [97]. Some implementations of tensor decomposition algorithms have been made publicly available, such as the N-way toolbox for MATLAB by Andersson and Bro [8] and the more recent MATLAB Tensor Toolbox by Bader and Kolda [11].

The Alternate Least Square (ALS) method was proposed in the original papers by Carroll and Chang [29] and Harshman [69] to realize the *CP Decomposition*, and it still remains the primary workhorse algorithm today due to its speed and ease of implementation [155]. Kang *et al.* conduct *CP Decomposition* on tensors with billions of sizes and hundreds of millions of nonzero elements by carefully minimizing the intermediate data size, and deploy tensor decompositions in the *MapReduce* framework [85]. Papalexakis *it al.* proposed a parallelizable method for speeding up *CP Decomposition* by leveraging random sampling techniques, which produce sparse factor matrices [132]

Tucker Decomposition is more flexible than *CP Decomposition*, although its application is usually limited by its limited scalability and vulnerability to noise. Memory-efficient Tucker decomposition [98] avoids constructing large intermediate results by handling the computation in a piecemeal fashion, adaptively selecting the order of operations. Tsourakakis proposed a scalable alternative using random sampling [159].

Several extensions have been developed to utilize tensor decomposition techniques on more practical applications. Tensor analysis has also been applied to study the dynamics of graphs and networks [149], which incrementally summarize tensor streams. Recent researches have further generalized the *CP Decomposition* to handle incomplete data [4]. Hayashi et al. [73] generalized the likelihood of *Tucker Decomposition* by using exponential-family distributions, and they proposed an anomaly detection approach by using the factor matrix as low-dimensional features of m-th mode of the tensor.

2.4.2 Non-negative Tensor Factorization

Non-negative tensor factorizations have been proposed to retain the non-negative characteristics of the original data [141], as natural expansions of non-negative matrix factorizations[102]. *Poisson Tensor Factorization* (*PTF*) [35] is one of such factorizations, and models *sparse count* data by describing the random variation via a Poisson distribution, such that each Poisson parameter is a multilinear combination of the model parameters.

2.4.3 Boolean Tensor Factorization

The Boolean Tensor Factorizations [114] factorize a binary tensor to binary factors using Boolean arithmetic, *i.e.*, defining that $1 + 1 = 1$. The problems of Boolean variations of *CP Decomposition* and *Tucker Decomposition* were proved to be NP-hard [115, 114], and algorithms for these problems were presented based on rather straight forward alternating optimization heuristics.

Chapter 3

Preliminaries

3.1 Definitions

In this section we describe the definitions used throughout this thesis.

3.1.1 Notations

Throughout, scalars are denoted by lowercase letters (a), vectors by boldface lowercase letters ($\mathbf{a}$), matrices by boldface capital letters ($\mathbf{A}$). The i-th element of a vector $\mathbf{a}$ is denoted by a_i, and the element of a matrix $\mathbf{A}$ with index (i,j) is denoted by a_{ij}. The i-th row vector of a matrix $\mathbf{A}$ is denoted by $\mathbf{a}_{i:}$, and the j-th column vector by $\mathbf{a}_{:j}$. The j-th column vector of a matrix $\mathbf{A}$ is also denoted by $\mathbf{a}_j$. The outer product of vector $\mathbf{a}$ and $\mathbf{b}$ is denoted by $\mathbf{a} \times \mathbf{b}$. The pseudoinverse of a matrix $\mathbf{A}$ is denoted by $\mathbf{A}^{\dagger}$, and the transposed matrix of $\mathbf{A}$ is denoted by $\mathbf{A}^T$.

A tensor is a multidimensional array. Its *order* is the dimensionality of the array, while each dimension is known as one *mode*. A first-order tensor is a vector, a second-order tensor is a matrix, and tensors of order three or higher are called higher-order tensors. A tensor is denoted by boldface Euler script letters ($\mathcal{A}$), and the *element* of $\mathcal{A}$ with index $(i_1, \ldots, i_M)$ by $a_{i_1,\ldots,i_M}$. We also use *measure* to denote the unit of each element. An order M tensor $\mathcal{A}$ of size $I_1 \times I_2 \times \ldots \times I_M$ is *rank-one* if it can be written as the outer product of M vectors, *i.e.*,

$$\mathcal{A} = \mathbf{x}^{(1)} \times \mathbf{x}^{(2)} \times \cdots \times \mathbf{x}^{(M)}. \tag{3.1}$$

This means that each element of the tensor is the product of the corresponding vector elements:

$$a_{i_1 i_2 \ldots i_M} = x^{(1)}_{i_1} x^{(2)}_{i_2} \cdots x^{(M)}_{i_M} \quad \text{for all} \quad (i_1, i_2, \ldots, i_M) \ . \tag{3.2}$$

The *rank* of a tensor is defined as the smallest number of rank-one tensors that can generate the tensor as their sum, and we refer to each rank-one tensor as a *component*. The notation $\|\mathcal{A}\|$ refers to the square root of the sum of the squares of the elements, analogous to the matrix Frobenius norm.

Table 3.1 and Table 3.2 list definitions of symbols used in this thesis.

Symbol	Definition
G	A simple graph or a bipartite graph
$\mathcal{G}$	A hypergraph
V	Nodes in a simple graph G or a hypergraph $\mathcal{G}$
V_1, V_2	Nodes of left and right partition in a bipartite graph G
$V_1, \ldots, V_k$	Nodes of 1st to kth partition in a k,k-hypergraph $\mathcal{G}$
I	The number of nodes in a simple graph G
I_1, I_2	The number of nodes of left and right partition in a bipartite graph G
$I_1, \ldots, I_k$	The number of nodes of 1st to kth partition in a k,k-hypergraph $\mathcal{G}$
E	Edges in a graph G
H	Hyperedges in a hypergraph $\mathcal{G}$

Table 3.1: Symbol table (continues to Table 3.2)

3.1.2 Basic Operation of Matrix and Tensor

Several matrix products are important in the sections that follow, so we briefly define them here. See details in [97].

Fibers are the higher-order analogue of matrix rows and columns. A fiber is defined by fixing every index but one. A matrix column is a mode-1 fiber and a matrix row is a mode-2 fiber. Third-order tensors have column, row, and tube fibers, denoted by $\mathbf{a}_{:jk}$, $\mathbf{a}_{i:k}$, and $\mathbf{a}_{ij:}$, respectively.

The mode-n matricization of a tensor $\mathcal{A}$ of size $I_1 \times I_2 \times \ldots \times I_M$ is denoted by $\mathbf{A}_{(n)}$ and arranges the mode-n fibers to be the columns of the resulting matrix. Though conceptually simple, the formal notation is clunky. Tensor element $(i_1, i_2, \ldots, i_M)$ maps to matrix element (i_n, j), where

$$j = 1 + \textstyle\sum_{k=1}^{n-1} \sum_{k=n+1}^{M} (i_k - 1) J_k \quad \text{with} \quad J_k = \prod_{m=1}^{n-1} \prod_{m=n+1}^{k-1} I_m. \quad (3.3)$$

See some examples in [97].

Symbol	Definition
$\mathbf{A}$	An adjacency matrix of a graph G
$\mathcal{A}$	An adjacency tensor of a k,k-hypergraph $\mathcal{G}$
a_{ij}	The element of $\mathbf{A}$ with index (i, j)
$a_{i_1 \dots i_M}$	The element of order M tensor $\mathcal{A}$ with index $(i_1, \dots, i_M)$
$\mathbf{a}_{i:}$	The ith row vector of $\mathbf{A}$
$\mathbf{a}_{:j}$	The jth column vector of $\mathbf{A}$
$\mathbf{A}_{(n)}$	The mode-n matricization of a tensor $\mathcal{A}$
λ_r	The rth largest eigenvalue of $\mathbf{A}$, singular value of $\mathbf{A}$, and weight of CP decomposition of $\mathcal{A}$
Λ	Diagonal matrix containing all λ_r
Λ_r	Diagonal matrix whose diagonal elements are $\lambda_1, \dots, \lambda_r$
$\mathbf{x}_r$	Eigenvector of $\mathbf{A}$ corresponding to λ_r
x_{ri}	i-th element of $\mathbf{x}_r$
$\mathbf{X}$	Matrix containing all eigenvectors $\mathbf{x}_r$ as their columns
$\mathbf{X}_r$	Matrix containing $\mathbf{x}_1, \dots, \mathbf{x}_r$ as its columns
$\mathbf{x}_r, \mathbf{y}_r$	Left and right singular vector of $\mathbf{A}$ corresponding to λ_r
x_{ri}, y_{ri}	i-th element of $\mathbf{x}_r, \mathbf{y}_r$
$\mathbf{X}, \mathbf{Y}$	Matrix containing all singular vectors $\mathbf{x}_r, \mathbf{y}_r$ as its columns
$\mathbf{X}_r, \mathbf{Y}_r$	Matrix containing $\mathbf{x}_1, \dots, \mathbf{x}_r$ and $\mathbf{y}_1, \dots, \mathbf{y}_r$ as their columns
$\mathbf{x}_r^{(m)}$	Factor vector of mth mode of $\mathcal{A}$ corresponding to λ_r
$x_{ri}^{(m)}$	i-th element of $\mathbf{x}_r^{(m)}$
$\mathbf{X}^{(m)}$	Factor matrix of mth mode containing all factor vectors $\mathbf{x}_r^{(m)}$ as its columns
$\mathbf{X}_r^{(m)}$	Matrix containing $\mathbf{x}_1^{(m)}, \dots, \mathbf{x}_r^{(m)}$ as their columns

Table 3.2: Symbol table (continued)

The *Kronecker product* of matrices $\mathbf{A}$ of size $I \times J$ and $\mathbf{B}$ of size $K \times L$ is denoted by $\mathbf{A} \otimes \mathbf{B}$. The result is a matrix of size $(IK) \times (JL)$ and defined by

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1J}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2J}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\mathbf{B} & a_{I2}\mathbf{B} & \cdots & a_{IJ}\mathbf{B} \end{bmatrix}. \tag{3.4}$$

This can be also written as

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_1 \otimes \mathbf{b}_2 & \mathbf{a}_1 \otimes \mathbf{b}_3 & \cdots & \mathbf{a}_J \otimes \mathbf{b}_{L-1} & \mathbf{a}_J \otimes \mathbf{b}_L \end{bmatrix}. \tag{3.5}$$

The *Khatri-Rao product* is the "matching columnwise" Kronecker product. Given matrices $\mathbf{A}$ of size $I \times J$ and $\mathbf{B}$ of size $J \times K$, their Khatri-Rao product is denoted by $\mathbf{A} \odot \mathbf{B}$. The result is a matrix of size $(IJ) \times K$ defined by

$$\mathbf{A} \odot \mathbf{B} = \begin{bmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_2 \otimes \mathbf{b}_2 & \cdots & \mathbf{a}_K \otimes \mathbf{b}_K \end{bmatrix}. \tag{3.6}$$

If $\mathbf{a}$ and $\mathbf{b}$ are vectors, then the Khatri-Rao and Kronecker products are identical, *i.e.*, $\mathbf{a} \otimes \mathbf{b} = \mathbf{a} \odot \mathbf{b}$.

The *Hadamard product* is the elementwise matrix product. Given matrices $\mathbf{A}$ and $\mathbf{B}$, both of size $I \times J$, their Hadamard product is denoted by $\mathbf{A} * \mathbf{B}$. The result is also of size $I \times J$ and defined by

$$\mathbf{A} * \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \cdots & a_{1J}b_{1J} \\ a_{21}b_{21} & a_{22}b_{22} & \cdots & a_{2J}b_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}b_{I1} & a_{I2}b_{I2} & \cdots & a_{IJ}b_{IJ} \end{bmatrix}. \tag{3.7}$$

3.1.3 Graph and k,k-Hypergraph

A simple graph which we call in this thesis is a set of nodes V where some pairs of nodes are connected by undirected edges. Formally, a simple graph is defined by $G =< V, E >$, where V is a set of all nodes and $E \subset V \times V$ is a set of all undirected edges. The *degree* (or *weighted degree*) of a node is the sum of the weights of the edges containing the node. In unweighted graph, it is the number of connected nodes.

A bipartite graph is a graph whose nodes are divided into two disjoint sets V_1 and V_2 such that every edge connects a node in V_1 to a node in V_2. Formally, a bipartite graph is defined by $G =< V_1 \cup V_2, E >$, where V_1 is a set of nodes of one partition, V_2 is a set of nodes of another partition, and $E \subset V_1 \times V_2$ is a set of all edges. Also, the *degree* (or *weighted degree*) of a node is the sum of the weights of the edges containing the node. In unweighted graph, it is the number of connected nodes.

A hypergraph is a generalization of a graph in which an edge can connect any number of nodes. Formally, a hypergraph is defined by $\mathcal{G} =< V, H >$, where V is a set of all nodes and H is a set of non-empty subset of V called hyperedges or simply edges. If all edges contain just k nodes, $\mathcal{G}$ is a k-uniform hypergraph. And a hypergraph is k-partite if V can be partitioned in k sets $V_1 \ldots V_k$ such that edges never contain multiple nodes from the same set. A k-partite, k-uniform hypergraph is called k,k-hypergraph. Thus, a k,k-hypergraph is defined by $\mathcal{G} =< V_1 \cup \ldots \cup V_k, H >$, where $V_1, \ldots, V_k$ are sets of nodes of each partition, and $H \subset V_1 \times \ldots \times V_k$ is a set of all hyperedges.

3.1.4 Adjacency Matrix and Adjacency Tensor

A simple graph or a bipartite graph can be represented as an adjacency matrix. In an adjacency matrix, columns and rows correspond to nodes, and values are 1s or *weights* if there exist edges between corresponding nodes, and 0 otherwise. An adjacency matrix of a simple graph is symmetric, because a simple graph is an undirected graph in our definition (Figure 1.1(a)), whereas that of a bipartite graph is not symmetric (Figure 1.1(b)).

A k,k-hypergraph can be represented as an adjacency tensor(Figure 1.1(c)). In an adjacency tensor, indices of each mode correspond to nodes of the corresponding partition, and values are 1s or *weights* if sets of nodes are contained by hyperedges, and 0 otherwise.

3.1.5 Community Structure

There are several definitions of community in graph data [139, 53]. In this thesis we use the term "*community*" as a subset of nodes in which almost all possible combinations of nodes have common edges.

In a simple graph, we regard so-called a near-clique as a *community.* A near-clique is a subset of nodes such that *almost every* two nodes in the subset are connected by an edge. Formally, a subset of nodes $< C >$ in a simple graph $G =< V, E >$ where $C \subset V$ is a near-clique if $|(C \times C) \cap E| \geq \epsilon|C \times C|$ with a specific value of ϵ. Nodes in a near-clique have similar corresponding row vectors $\mathbf{a}_{i:}$ in the adjacency matrix $\mathbf{A}$.

A *community* in a bipartite graph in this thesis is so-called a near-bipartite core. A near-bipartite core is a subset of nodes such that *almost every* two nodes of different partitions in the subset are connected by an edge. Formally, a subset of nodes $< C_1 \cup C_2 >$ in a bipartite graph $G =< V_1 \cup V_2, E >$ where $C_1 \subset V_1, C_2 \subset V_2$ is a near-bipartite core if $|(C_1 \times C_2) \cap E| \geq \epsilon|C_1 \times C_2|$ with a specific value of ϵ. Nodes in a near-bipartite core have similar corresponding row vectors $\mathbf{a}_{i:}$ or column vectors $\mathbf{a}_{:j}$ in the adjacency matrix $\mathbf{A}$.

Also, we define a k,k-hyperclique as a *community* in a k,k-hypergraph. A k,k-hyperclique is a subset of nodes such that *almost every* combinations of

k nodes straddling over k partitions share common hyperedges. Formally, a subset of nodes $< C_1 \cup \ldots \cup C_k >$ in a k,k-hypergraph $\mathcal{G} =< V_1 \cup \ldots \cup V_k, H >$ where $C_1 \subset V_1, \ldots, C_k \subset V_k$ is a k,k-hyperclique if $|(C_1 \times \ldots \times C_k) \cap H| \geq \epsilon |C_1 \times \ldots \times C_k|$ with a specific value of ϵ. In an adjacency tensor, $(k-1)$ dimensional sections corresponding to nodes of a specific partition in a k,k-hyperclique resemble each other.

3.2 Matrix and Tensor Decomposition

Eigen decomposition, singular value decomposition, and tensor decomposition are basic techniques we use in this thesis. We can commonly understand these techniques in view of low-rank approximation of matrices and tensors, and the scores of each nodes calculated by these techniques have similar relational expressions. Here we describe a brief introduction of these techniques, along with the common properties that these techniques have.

3.2.1 Eigen Decomposition

We consider a low-rank approximation of a symmetric matrix $\mathbf{A}$ of size $I \times I$ so as to minimize the Frobenius norm

$$\left\| \mathbf{A} - \sum_{r=1}^{R} \lambda_r \left(\mathbf{x}_r \times \mathbf{x}_r \right) \right\|,$$

where R is desired rank, λ_r is a scalar ($\lambda_1 \geq \ldots \geq \lambda_R$), and $\mathbf{x}_r$ is rth unit vector of size I (Figure 3.1(a)). The solution is obtained in terms of eigen decomposition, *i.e.*, $\lambda_1, \ldots, \lambda_R$ are R largest eigenvalues of $\mathbf{A}$ and $\mathbf{x}_r$ is the r-th eigenvector of $\mathbf{A}$ corresponding to λ_r.

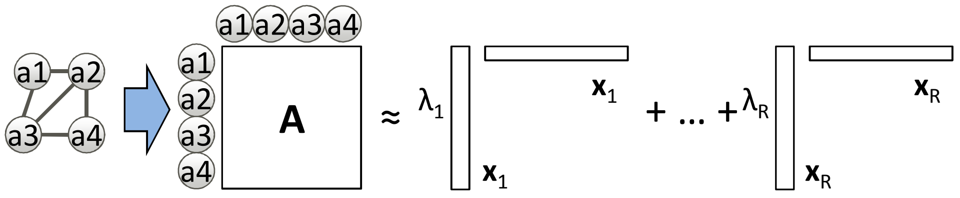
Because λ_r and $\mathbf{x}_r$ are eigenvalues and eigenvectors, they satisfy

$$\mathbf{A}\mathbf{x}_r = \lambda_r \mathbf{x}_r. \tag{3.8}$$

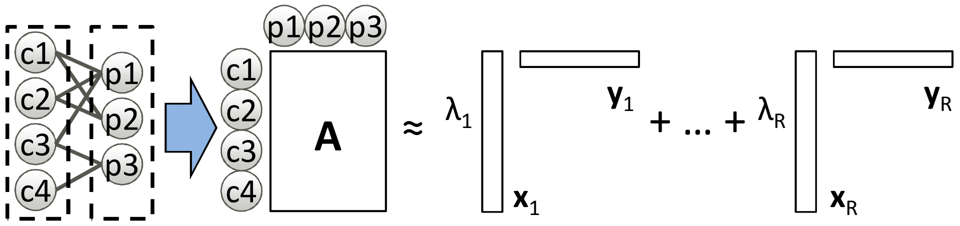
For an adjacency matrix of a simple graph, this can be written as

$$x_{ri} = \lambda_r^{-1} \sum_{j \in C_i} a_{ij} x_{rj}, \tag{3.9}$$

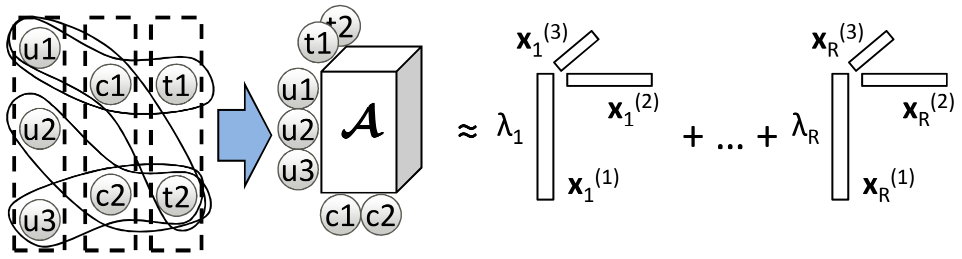
where C_i is a set of nodes connected to i-th node, and a_{ij} is *weight* of the edge between i-th node and j-th node. In this thesis we call x_{ri} the r-th *eigenscore* (or just *score*) of the i-th node.



(a) eigen decomposition



(b) singular value decomposition (SVD)



(c) CANDECOMP / PARAFAC (CP) decomposition

Figure 3.1: Low-rank approximation of matrix and tensor.

3.2.2 Singular Value Decomposition

For a matrix $\mathbf{A}$ which is not symmetric, we consider a low-rank approximation of $\mathbf{A}$ of size $I_1 \times I_2$ so as to minimize the Frobenius norm

$$\left\| \mathbf{A} - \sum_{r=1}^{R} \lambda_r \left(\mathbf{x}_r \times \mathbf{y}_r \right) \right\|,$$

where R is desired rank, λ_r is a scalar ($\lambda_1 \geq \ldots \geq \lambda_R$), $\mathbf{x}_r$ is r-th unit vector of size I_1, and $\mathbf{y}_r$ is r-th unit vector of size I_2 (Figure 3.1(b)). The solution is obtained in terms of singular value decomposition (SVD), *i.e.*, $\lambda_1, \ldots, \lambda_R$ are R largest singular values of $\mathbf{A}$, $\mathbf{x}_r$ is the r-th left singular vector of $\mathbf{A}$ corresponding to λ_r, and $\mathbf{y}_r$ is the r-th right singular vector of $\mathbf{A}$ corresponding to λ_r.

Because λ_r and $\mathbf{x}_r$, $\mathbf{y}_r$ are singular values and singular vectors, they satisfy

$$\begin{cases} \mathbf{A}\mathbf{x}_r = \lambda_r \mathbf{y}_r, \\ \mathbf{A}\mathbf{y}_r = \lambda_r \mathbf{x}_r. \end{cases} \tag{3.10}$$

For an adjacency matrix of a bipartite graph, this can be written as

$$\begin{cases} x_{ri} = \lambda_r^{-1} \sum_{j \in C_i} a_{ij} y_{rj}, \\ y_{rj} = \lambda_r^{-1} \sum_{i \in D_j} a_{ij} x_{ri}, \end{cases} \tag{3.11}$$

where C_i is a set of right nodes connected to i-th left node, D_j is a set of left nodes connected to j-th right node, and a_{ij} is *weight* of the edge between i-th left node and j-th right node. In this thesis we call x_{ri} the r-th left *singular score* of the i-th left node, and y_{rj} the r-th right *singular score* of the j-th right node (or just *score*).

3.2.3 Tensor Decomposition

For a tensor $\mathcal{A}$, we consider a low-rank approximation of $\mathcal{A}$ of size $I_1 \times \ldots \times I_M$ so as to minimize the Frobenius norm

$$\left\| \mathcal{A} - \sum_{r=1}^{R} \lambda_r \left(\mathbf{x}_r^{(1)} \times \ldots \times \mathbf{x}_r^{(M)} \right) \right\|,$$

where R is desired rank, λ_r is a scalar ($\lambda_1 \geq \ldots \geq \lambda_R$), $\mathbf{x}_r^{(1)}, \ldots, \mathbf{x}_r^{(M)}$ is r-th unit vectors of size $I_1, \ldots, I_M$ (Figure 3.1(c)). The solution is obtained in terms of CANDECOMP / PARAFAC (CP) decomposition [97]. The matrix $\mathbf{X}^{(m)} = \left[\mathbf{x}_1^{(m)} \cdots \mathbf{x}_R^{(m)} \right]$ are called *factor matrix*. In this thesis we call each vector $\mathbf{x}_r^{(m)}$ as a *factor vector*, and λ_r as a r-th *weight*.

Because CP decomposition is not a problem with orthogonality constraints, factor vectors do not satisfy rather simple relational expression as eigen decomposition and singular value decomposition. Even though, in a solution of the three-order tensor in which the sum of rank-one tensors fits the original tensor $\mathcal{A}$ perfectly, each factor matrix satisfies [97]

$$\begin{cases} \mathbf{X}^{(1)} = \mathbf{A}_{(1)}(\mathbf{X}^{(2)} \odot \mathbf{X}^{(3)})(\mathbf{X}^{(2)T}\mathbf{X}^{(2)} * \mathbf{X}^{(3)T}\mathbf{X}^{(3)})^{\dagger}\Lambda^{\dagger}, \\ \mathbf{X}^{(2)} = \mathbf{A}_{(2)}(\mathbf{X}^{(1)} \odot \mathbf{X}^{(3)})(\mathbf{X}^{(1)T}\mathbf{X}^{(1)} * \mathbf{X}^{(3)T}\mathbf{X}^{(3)})^{\dagger}\Lambda^{\dagger}, \\ \mathbf{X}^{(3)} = \mathbf{A}_{(3)}(\mathbf{X}^{(1)} \odot \mathbf{X}^{(2)})(\mathbf{X}^{(1)T}\mathbf{X}^{(1)} * \mathbf{X}^{(2)T}\mathbf{X}^{(2)})^{\dagger}\Lambda^{\dagger}, \end{cases} \tag{3.12}$$

where Λ is a diagonal matrix whose diagonal elements are weights of CP decomposition. For an adjacency tensor of k,k-hypergraph, this can be written as

$$\begin{cases} x^{(1)}_{ri_1} = \lambda_r^{-1} \displaystyle\sum_{(i_2,i_3)\in C^{(1)}_{i_1}} a_{i_1 i_2 i_3} \sum_{s=1}^{S} p^{(1)}_{rs} x^{(2)}_{si_2} x^{(3)}_{si_3}, \\ x^{(2)}_{ri_2} = \lambda_r^{-1} \displaystyle\sum_{(i_1,i_3)\in C^{(2)}_{i_2}} a_{i_1 i_2 i_3} \sum_{s=1}^{S} p^{(2)}_{rs} x^{(1)}_{si_1} x^{(3)}_{si_3}, \\ x^{(3)}_{ri_3} = \lambda_r^{-1} \displaystyle\sum_{(i_1,i_2)\in C^{(3)}_{i_3}} a_{i_1 i_2 i_3} \sum_{s=1}^{S} p^{(3)}_{rs} x^{(1)}_{si_1} x^{(2)}_{si_2}, \end{cases} \tag{3.13}$$

where S is the rank of $\mathcal{A}$, and $C^{(m)}_{i_m}$ is a set of combination of nodes which share common hyperedges with i_m-th node in m-th partition. $p^{(1)}_{rs}$ is the (r, s)-th element of $(\mathbf{X}^{(2)T}\mathbf{X}^{(2)} * \mathbf{X}^{(3)T}\mathbf{X}^{(3)})^{\dagger}$, and this is the same for $p^{(2)}_{rs}$ and $p^{(3)}_{rs}$. This equation suggests that values of elements of factor vectors corresponding to small weights can affect those corresponding to large weights. This discussion can be easily generalized to a tensor with more than three orders. In this thesis we call $x^{(m)}_{ri_m}$ the r-th *eigenscore* of the i_m-th node of m-th mode (or just *score*).

3.2.4 Common Properties of Decompositions

As described above, eigen decomposition, singular value decomposition and CP decomposition have similar relational expressions such as equations 3.9, 3.11, and 3.13. Roughly speaking, all of these equations suggest that a value of r-th vector corresponding to a node is linear combination of values corresponding to connected nodes multiplied by the inverse of λ_r, and values decay exponentially with the decay rate of λ_r if nodes are connected with a few edges. In this section we show how real-world datasets have this property by using a 1-hour snapshot of LBNL traffic logs (LBNL) described in Section 3.3. LBNL traffic logs datasets are sets of packet traces which include four attributes: IP-source (srcIPs), IP-destination (dstIPs), port number (ports), and a time tick in second.

To discuss eigen decomposition, we construct a simple graph whose nodes are dstIPs and connected if they share the same port. Eigenvectors are calculated for a symmetric adjacency matrix in which each element is 1 if the corresponding node pair is connected, and 0 otherwise. We plot edges by the

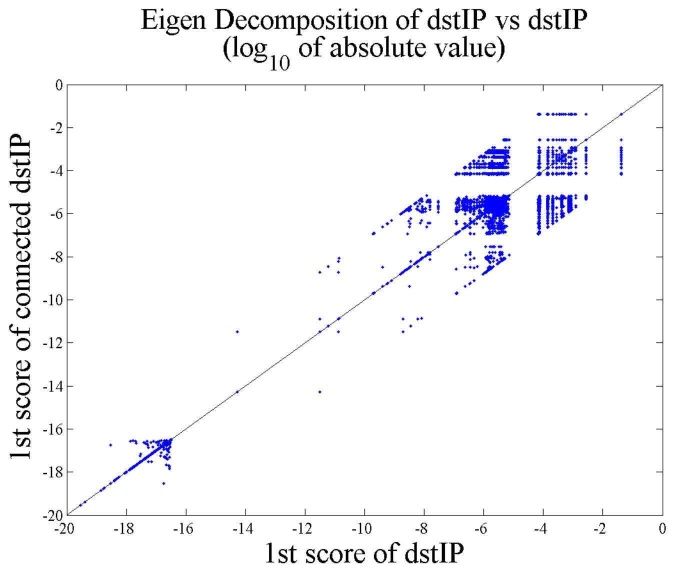


Figure 3.2: Edges are plotted by absolute values of 1st eigenscores of two nodes on the edges. The adjacency matrix is related to a simple graph whose nodes are dstIPs and connected if they share same ports in a LBNL traffic logs dataset.

absolute values of the 1st eigenscores corresponding to nodes of one side of each edge against those corresponding to nodes of another side (Figure 3.2). The largest eigenvalue of this adjacency matrix is 603.1, *i.e.*, $10^{2.8}$, and this plot suggests that the ratio of absolute values of two eigenscores of nodes of the same edge does not exceed corresponding eigenvalue. How true is this for many real-world graph data? The equation 3.9 can be written as

$$|x_{ri}| = |\lambda_r|^{-1} \sum_{j \in C_i} a_{ij} |x_{rj}|, \tag{3.14}$$

if the eigenscores x_{rj} for $j \in C_i$ are all positive or all negative. In this case, $a_{ij}|x_{rj}|/|x_{ri}|$ is not greater than $|\lambda_r|$ for all $j \in C_i$. Conversely, if the eigenscores x_{ri} for $i \in C_j$ are all positive or all negative, $a_{ij}|x_{ri}|/|x_{rj}|$ is not greater than $|\lambda_r|$ for all $i \in C_j$. Thus, if two connected nodes are connected to the nodes with eigenscores of all positive or all negative, the ratio of the absolute values of the eigenscores of these two nodes does not exceed the absolute value of the eigenvalue. A typical graph structure is a "caveman graph" where a node in a group of nodes (a "cave") is connected to many other nodes in the same cave, while connected to very little nodes in different caves [83]. From equation 3.9, the absolute value of the eigenscore of a node i in a cave S satisfies

$$|x_{ri}| \leq |\lambda_r|^{-1}(a_{ig}|x_{rg}| + \sum_{\substack{j \in C_i \\ j \neq g}} a_{ij}|x_{rj}|) \tag{3.15}$$

$$\leq |\lambda_r|^{-1} a_{max}(|x_{rg}| + \sum_{\substack{j \in S \\ j \neq g}} |x_{rj}|), \tag{3.16}$$

where node g is a node connected to the nodes outside the cave (say, a "gatekeeper'), a_{max} is a maximum value of $a_{ij}(i \in S, j \in S)$. By summing up the inequation 3.16 for all $i \in S(i \neq g)$, we obtain an inequation

$$(|\lambda_r| - a_{max}(|S| - 1)) \sum_{\substack{j \in S \\ j \neq g}} |x_{rj}| \leq a_{max}(|S| - 1)|x_{rg}|. \tag{3.17}$$

The inequation 3.17 indicates that the absolute values of the eigenscores of all nodes in a cave S except for the gatekeeper are much smaller than that of the gatekeeper if $a_{max}(|S| - 1)$ is much smaller than $|\lambda_r|$. Along with the equation 3.9, the eigenscores of the nodes in "caveman" structures are all positive or all negative, that is, the same sign as that of the gatekeeper. Thus, for many nodes in a graph which has the "caveman" structure, the ratio of absolute values of two eigenscores of nodes of the same edge does not exceed corresponding eigenvalue. This means that nodes residing in close neighborhood in a graph have similar eigenscores. In other words, eigenscores tend to decay exponentially as nodes are separated from a node with the highest eigenscore, such that the decay rate should be about corresponding eigenvalue. This is the same as other eigenvectors, and we would be able to know distance between nodes by using several eigenvectors. Based on this intuition, we propose a novel method to estimate the shortest path distance of a given node pair by using eigenvalues and eigenvectors in Chapter 6.

Next we construct a bipartite graph whose nodes are dstIPs and ports, to discuss singular value decomposition. Singular vectors are calculated for an adjacency matrix in which each element is 1 if corresponding node pair is connected, and 0 otherwise. We plot edges by the absolute values of the left 1st singular scores corresponding to dstIPs against those of the right 1st singular scores corresponding to ports (Figure 3.3). The largest singular value of this adjacency matrix is 122.7, *i.e.*, $10^{2.1}$, and this plot suggests that the ratio of absolute values of the left singular score and the right singular

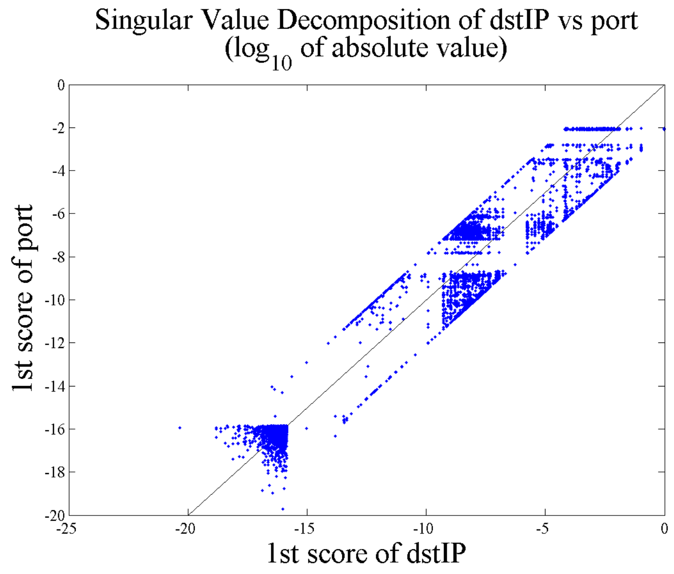
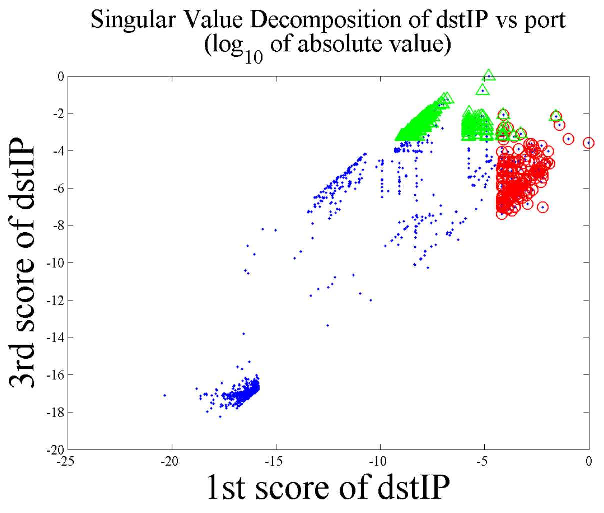


Figure 3.3: Edges are plotted by absolute values of 1st singular scores of two nodes on the edges. The adjacency matrix is related to a bipartite graph whose nodes are dstIPs and ports in a LBNL traffic logs dataset.

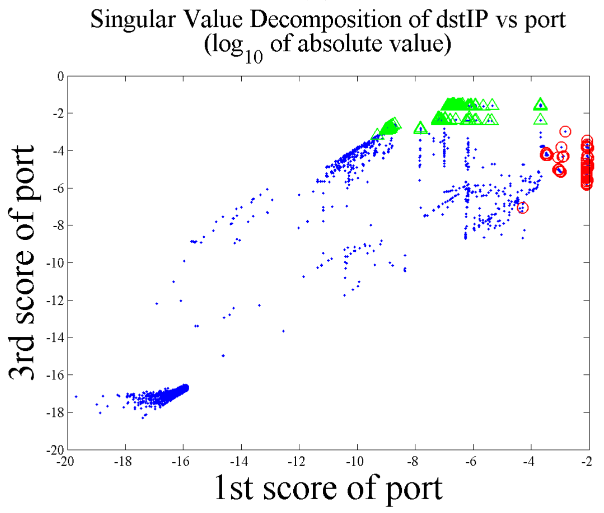
score of a node pair on the same edge does not exceed corresponding singular value, similar to eigen decomposition. We can discuss how true this is for many real-world graph data as same as the case of eigen decomposition. The equation 3.11 can be written as

$$\begin{cases} |x_{ri}| = |\lambda_r|^{-1} \sum_{j \in C_i} a_{ij} |y_{rj}|, \\ |y_{rj}| = |\lambda_r|^{-1} \sum_{i \in D_j} a_{ij} |x_{ri}| \end{cases} \tag{3.18}$$

if the singular scores x_{ri} for $i \in D_j$ and y_{rj} for $j \in C_i$ are all positive or all negative. In this case, $a_{ij}|y_{rj}|/|x_{ri}|$ is not greater than $|\lambda_r|$ for all $j \in C_i$, and $a_{ij}|x_{ri}|/|y_{rj}|$ is not greater than $|\lambda_r|$ for all $i \in D_j$. Thus, if two connected nodes are connected to the nodes with singular scores of all positive or all negative, the ratio of the absolute values of the singular scores of these two nodes does not exceed the absolute value of the singular value. Similar to the case of eigen decomposition, a typical graph structure is a "caveman graph". From equation 3.11, the absolute value of the singular score of a



(a)



(b)

Figure 3.4: Nodes are plotted by absolute values of 1st singular scores against those of 3rd singular scores. The adjacency matrix is related to a bipartite graph whose nodes are dstIPs and ports in a LBNL traffic logs dataset. Nodes share same nodes of opposite partition with a node of the highest 1st (3rd) singular score are plotted in red circle (green triangle). (a) nodes of dstIPs, (b) nodes of ports.

left node $i \in S_1$ and a right node $j \in S_2$ in a cave $S_1 \cup S_2$ satisfies

$$\begin{cases} |x_{ri}| \leq |\lambda_r|^{-1} a_{max}(|y_{rg}| + \sum_{\substack{j \in S_2 \\ j \neq g}} |y_{rj}|), \\ |y_{rj}| \leq |\lambda_r|^{-1} a_{max} \sum_{i \in S_1} |x_{ri}|, \end{cases} \tag{3.19}$$

where node g is a gatekeeper, a_{max} is a maximum value of $a_{ij}(i \in S_1, j \in S_2)$. We assume that the gatekeeper is a node in the right partition. By summing up the inequation 3.19 for all $i \in S_1$ and $j \in S_2 (j \neq g)$, we obtain an inequation

$$(|\lambda_r| - a_{max}(|S_2| - 1)) \sum_{i \in S_1} |x_{ri}| + (|\lambda_r| - a_{max}|S_1|) \sum_{\substack{j \in S_2 \\ j \neq g}} |y_{rj}| \leq a_{max}|S_1||y_{rg}|. \tag{3.20}$$

The inequation 3.20 indicates that the absolute values of the singular scores of all nodes in a cave $S_1 \cup S_2$ except for the gatekeeper are much smaller than that of the gatekeeper if $a_{max}|S_1|$ and $a_{max}(|S_2| - 1)$ are much smaller than $|\lambda_r|$. Along with the equation 3.11, the singular scores of the nodes in "caveman" structures are all positive or all negative, that is, the same sign as that of the gatekeeper. Thus, for many nodes in a graph which has the "caveman" structure, the ratio of absolute values of the left singular score and the right singular score of a node pair on the same edge does not exceed corresponding singular value, similar to eigen decomposition. This means that nodes residing in close neighborhood in a graph have similar singular scores, like eigenscores, and also indicates that singular scores of nodes in a same community structure, in which every node pairs reside in close location, should form a cluster of similar values.

Moreover, we plot nodes by the absolute values of the 1st singular scores against those of the 3rd singular scores (Figure 3.4). The third largest singular value of this adjacency matrix is 42.5, *i.e.*, $10^{1.6}$. Red circle or green triangle nodes are nodes which have 2-hops paths to a node of the highest 1st or 3rd singular score. The 1st singular scores of red circle nodes are much larger than those of green triangle nodes, whereas the 3rd singular scores of green triangle nodes are much larger than those of red circle nodes. Because a left singular score is the sum of the right singular scores of the connected nodes (equation 3.11), the 1st singular scores of dstIP nodes should be correlated to the number of connected red port nodes. This is the same for the 1st singular scores of port nodes, and 3rd singular scores. If red circle nodes and green triangle nodes form community structures, these singular scores should be correlated to the number of connections with nodes of each community. In Chapter 5, we propose a novel method of pattern discovery based on this intuition, which spots *patterns* related to community structures effectively by analyzing distribution of singular scores and degrees of nodes.

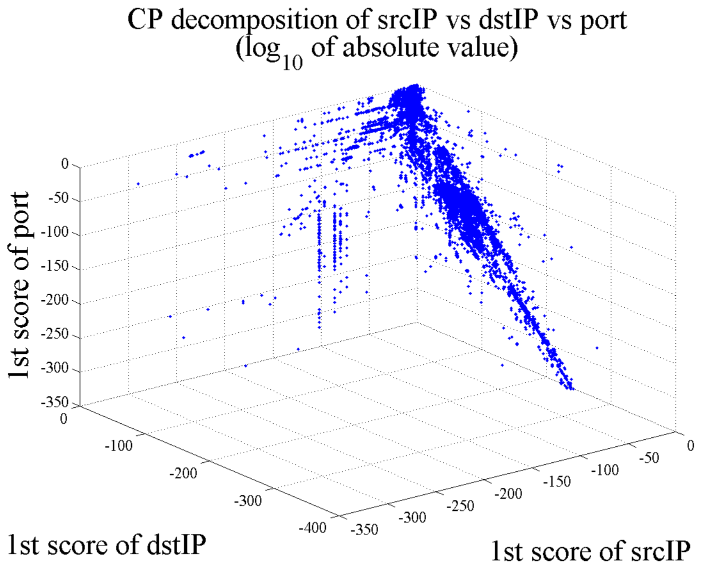


Figure 3.5: Hyperedges are plotted by absolute values of 1st eigenscores of nodes on the hyperedges. The adjacency tensor is related to a k,k-hypergraph whose nodes are srcIPs, dstIPs, and ports in a LBNL traffic logs dataset.

Finally we construct a k,k-hypergraph whose nodes are srcIPs, dstIPs, and ports, to discuss CP decomposition. Function vectors are calculated for an adjacency tensor in which each element is 1 if corresponding combination of nodes are on a hyperedge, and 0 otherwise. The highest weight of the CP decomposition is 26.9, *i.e.*, $10^{1.4}$. Figure 3.5 is a 3D-plots of hyperedges plotted by absolute values of the 1st eigenscores of nodes on hyperedges. This implementation of CP decomposition can calculate the values of element of factor matrix with precision of about 10^{-300}. We can see that, for most of the hyperedges, the ratio of eigenscores of a node of dstIP and that of port on the same hyperedge does not exceed a specific rate, whereas the eigenscore of a node of srcIP on the same hyperedge does not depend on those

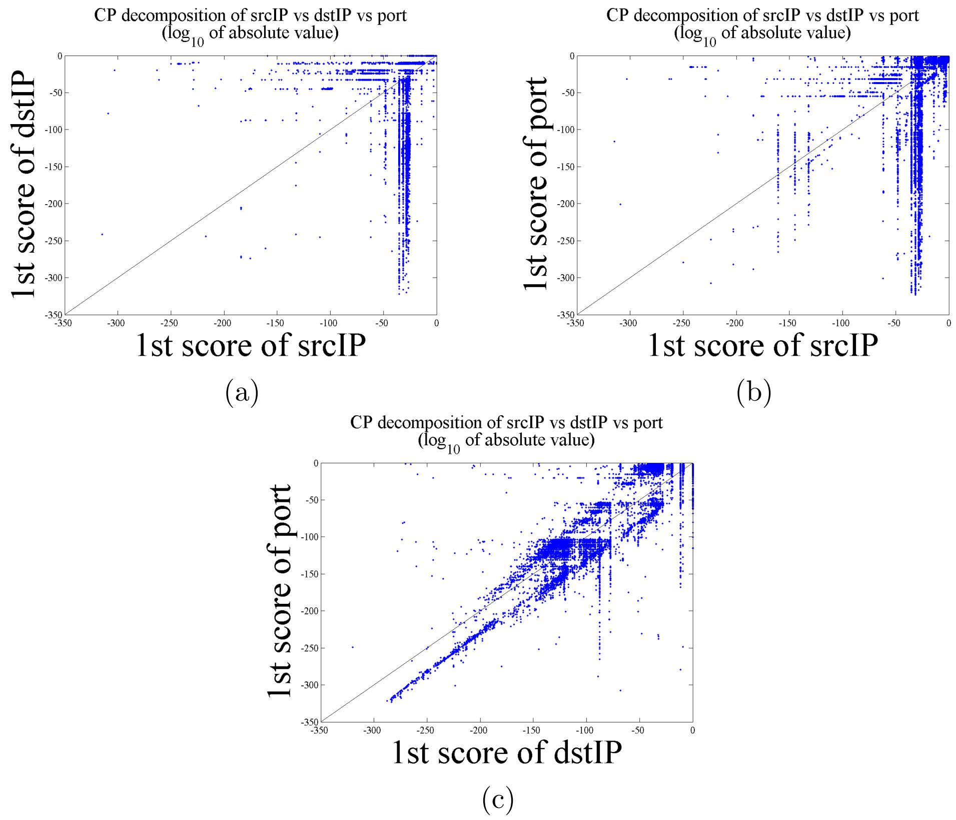


Figure 3.6: Hyperedges of the same k,k-hypergraph as Figure 3.5 are plotted by 2D-plots. (a) srcIPs vs dstIPs, (b) srcIPs vs ports, and (c) dstIPs vs ports.

of dstIP and port. To understand that more clearly, we also show 2D-plots plotted by values of two of three nodes on hyperedges (Figure 3.6(a)(b)(c)). The ratio of eigenscores of a node of dstIP and that of port on the same hyperedge is within a specific range (Figure 3.6(c)), whereas eigenscores of nodes of srcIP are high on most of edges, not depending on those of dstIP and port (Figure 3.6(a)(b)). This is analogous to eigen decomposition and singular value decomposition, meaning that nodes resided in close location in hypergraph have similar eigenscores, in at least some modes. In other words, in this dataset, eigenscores of CP decomposition decay exponentially as nodes are separated from a node with the highest eigenscore, in these modes. Is this true in many other k,k-hypergraphs? This is very hard to prove, but similar to eigen decomposition and singular value decomposition, the equation 3.13 can be written as

$$\begin{cases} \left|x_{ri_1}^{(1)}\right| = |\lambda_r|^{-1} \displaystyle\sum_{(i_2,i_3)\in C_{i_1}^{(1)}} a_{i_1i_2i_3} \sum_{s=1}^{S} \left|p_{rs}^{(1)}x_{si_2}^{(2)}x_{si_3}^{(3)}\right|, \\ \left|x_{ri_2}^{(2)}\right| = |\lambda_r|^{-1} \displaystyle\sum_{(i_1,i_3)\in C_{i_2}^{(2)}} a_{i_1i_2i_3} \sum_{s=1}^{S} \left|p_{rs}^{(2)}x_{si_1}^{(1)}x_{si_3}^{(3)}\right|, \\ \left|x_{ri_3}^{(3)}\right| = |\lambda_r|^{-1} \displaystyle\sum_{(i_1,i_2)\in C_{i_3}^{(3)}} a_{i_1i_2i_3} \sum_{s=1}^{S} \left|p_{rs}^{(3)}x_{si_1}^{(1)}x_{si_2}^{(2)}\right|, \end{cases} \tag{3.21}$$

if the values $p_{rs}^{(1)}x_{si_2}^{(2)}x_{si_3}^{(3)}$ for $(i_2, i_3) \in C_{i_1}^{(1)}$ and $p_{rs}^{(2)}x_{si_1}^{(1)}x_{si_3}^{(3)}$ for $(i_1, i_3) \in C_{i_2}^{(2)}$ and $p_{rs}^{(3)}x_{si_1}^{(1)}x_{si_2}^{(2)}$ for $(i_1, i_2) \in C_{i_3}^{(3)}$ are all positive or all negative for all $s = 1, \ldots, S$. In this case,

$$\begin{cases} a_{i_1i_2i_3} \left|p_{rs}^{(1)}x_{si_2}^{(2)}x_{si_3}^{(3)}\right| / \left|x_{ri_1}^{(1)}\right| \le |\lambda_r| \quad \forall(i_2, i_3) \in C_{i_1}^{(1)}, \\ a_{i_1i_2i_3} \left|p_{rs}^{(2)}x_{si_1}^{(1)}x_{si_3}^{(3)}\right| / \left|x_{ri_2}^{(2)}\right| \le |\lambda_r| \quad \forall(i_1, i_3) \in C_{i_2}^{(2)}, \\ a_{i_1i_2i_3} \left|p_{rs}^{(3)}x_{si_1}^{(1)}x_{si_2}^{(2)}\right| / \left|x_{ri_3}^{(3)}\right| \le |\lambda_r| \quad \forall(i_1, i_2) \in C_{i_3}^{(3)}, \end{cases} \tag{3.22}$$

for $s = 1, \ldots, S$. If $\left|x_{si_3}^{(3)}\right|$ are similar values on most of edges like srcIP nodes in Figure 3.6(a)(b), $\left|x_{si_2}^{(2)}\right| / \left|x_{ri_1}^{(1)}\right|$ and $\left|x_{si_1}^{(1)}\right| / \left|x_{ri_2}^{(2)}\right|$ might be not greater than a specific value. These assumptions are hard to translate into structures in real-world graph data, but we observed almost similar patterns as in Figure 3.5, in most of the datasets we analyzed in this thesis. According to this observation, we might expect that groups of nodes which are contained by hyperedges containing similar nodes, *i.e.*, community structures in k,k-hypergraph, should have eigenscores of similar order in at least some modes, *i.e.*, similar number of digit, and be able to be separated if we sort nodes according to the order of the eigenscores. Based on this intuition, we propose a novel method to detect community structures in k,k-hypergraph that we call k,k-hyperclique in Chapter 4.

3.3 Datasets Used

In this section we describe the various real-world datasets that we have used in this thesis. Some of datasets such as social networks are fundamentally a

Dataset name	**abbr.**	**attributes**		**chap.**
LBNL traffic logs	LBNL	2.3K	IP-srcs	4,5
		2.4K	IP-dsts	
		6.1K	port #'s	
		3.6K	time ticks	
Read the Web	RTW	3.6K	subjects	4
		0.1K	verbs	
		3.9K	objects	
Berkeley Drosophila Genome Project	BDGP	4.5K	genes	4
		0.2K	terms	
		6	stages	
The Internet Movie Database	IMDb	553.0K	actors	5,6
		204.0K	movies	
DBLP collaboration network	DBLP	316.0K	authors	5,6
		472.0K	papers	
Patent Citation	Patent	2,089.0K	citing	5
		3,259.0K	cited	
Click Stream	Click	23.0K	users	5
		199.0K	sites	

Table 3.3: A summary of datasets (continues to Table 3.4)

simple graph in which nodes are not partitioned, and we mainly use them in Chapter 6. Others are datasets which can be regarded as bipartite graphs or k,k-hypergraphs. In Chapter 5 we pick up two attributes of each dataset and analyze them as bipartite graphs, and in Chapter 4 we use more than two attributes and analyze them as k,k-hypergraphs. Moreover, we pick up one attribute of some datasets and analyze them as a simple graph with no partition in Chapter 6, in which there is an edge between two nodes if they share same values in some other attributes. Summaries of all datasets are shown in Table 3.3 and Table 3.4.

LBNL traffic logs (LBNL) The network traffic log is made available through a research effort to study the characteristics of traffic for Internet enterprises [129]. The measurement was taken on servers within the Lawrence Berkeley National Lab (LBNL) from thousands of internal hosts over time, with millions of packet traces recorded[1]. Each packet

[1]http://www.icir.org/enterprise-tracing/download.html

Dataset name	**abbr.**	**attributes**	**chap.**
The CAIDA AS Relationships Datasets	AS-CAIDA	26.5K AS	6
Email communication network from Enron	Enron	36.7K address	6
Arxiv Astro Physics	AstroPh	18.8K authors	6
Arxiv Condensed Matter	CondMat	23.1K authors	6
Arxiv General Relativity	GrQc	5.2K authors	6
Arxiv High Energy Physics Theory	HepTh	9.9K authors	6

Table 3.4: A summary of datasets (continued)

trace includes four attributes: IP-source, IP-destination, port number, and a time tick in second. With privacy in concern, lower 16 bits of IP addresses were randomly permuted to anonymize the host identity, whereas upper 16 bits were kept intact for proper identification of the location and service provider [130]. This dataset consists of 264 logs, each log is a snapshot of 1-hour. Moreover, logs are separated by two kinds of port number, *i.e.*, TCP and UDP, and logs with or without scanning activity (we call 'scanners' or 'regular' dataset). We use three or four attributes in Chapter 4, and two attributes in Chapter 5. Also, we use only logs of regular dataset with TCP port number in Chapter 4.

Read the Web (RTW) This online knowledge base is the outcome of the NELL (Never-Ending Language Learning) system at Carnegie Mellon University [27]. It employs natural language processing and machine learning techniques to constantly and automatically crawl web pages and extract facts [28]. Each fact is a triplet of (subject, verb, object) such as (*pittsburgh*, *city-located-in-state*, *pennsylvania*). For better quality of results, we apply our algorithm on a preprocessed subset after noise removal (by courtesy of Dr. Byran Kisiel at Carnegie Mellon University).

Berkeley Drosophila Genome Project (BDGP) The dataset was collected from the Berkeley Drosophila Genome Project (BDGP) to study the spatial-temporal patterns of gene expression during the early development of fruit fly [153, 154]. In Chapter 4 we select three attributes from the database dump available at [1], which consists of genes, functional annotation terms from a specialized vocabulary, and different developmental stages.

The Internet Movie Database (IMDb) The Internet Movie Data Base (IMDb[2]) is a collection of facts about movies and actors. We analyze as a bipartite graph of movies and actors in Chapter 5, and as a simple graph of a collaboration network between actors in Chapter 6.

DBLP collaboration network (DBLP) DBLP[3] provides bibliographic information on major computer science journals and proceedings. We analyze as a bipartite graph of authors and papers in Chapter 5, and as a simple graph of a collaboration network between authors in Chapter 6.

Patent Citation(Patent) Patent citation data between citing patents and cited patents[67].

Click Stream(Click) Click stream data about people clicked link of sites[118]. This dataset contains active time of each session.

The CAIDA AS Relationships Datasets (AS-CAIDA) CAIDA AS graph from November 5 2007. CAIDA AS is a communication network of who-talks-to-whom constructed from the BGP (Border Gateway Protocol) logs of Autonomous Systems (AS). The graph of routers comprising the Internet can be organized into sub-graphs called Autonomous Systems.

Email communication network from Enron (Enron) Email communication network in a company. The nodes of the network are email addresses, and two nodes are connected with an edge if one address sent at least one email to the other address.

Arxiv Astro Physics (AstroPh) Scientific collaborations in astro physics.

Arxiv Condensed Matter (CondMat) Scientific collaborations in condense matter physics.

[2]http://www.imdb.com

[3]http://dblp.uni-trier.de/

Arxiv General Relativity (GrQc) Scientific collaborations in general relativity and quantum cosmology.

Arxiv High Energy Physics Theory (HepTh) Scientific collaborations in high energy physics - theory.

The datasets AS-CAIDA, Enron, AstroPh, CondMat, GrQc, and HepTh were taken from the Stanford Network Analysis Project (SNAP[4]).

[4]http://snap.stanford.edu/

Chapter 4

Anomaly Detection Using Tensor Decomposition

A lot of modern applications such as web knowledge bases, network traffic monitoring and online social bookmarks involve several kind of relational data, which are sometimes called 'heterogeneous' network data in data mining research communities, with rich types of interactions among nodes. Many of them can be analyzed as k,k-hypergraphs as discussed in Chapter 1. Suspicious activities such as spam bookmarks and distributed cyber attacks tend to form clique-like patterns of a k,k-hypergraph, in which almost every combinations of k nodes straddling over k partitions share common hyperedges. Moreover, their nodes are hardly related to the *important* nodes, *i.e.*, they have few l-hops paths of small l to the *important* nodes which have many l-hops paths of small l to many nodes in the graph. An important and non-trivial problem is to detect such clique-like patterns in a k,k-hypergraph with millions of edges in a scalable way, and distinguish them from those caused by NOT suspicious behaviors, such as many popular contents bookmarked by many users with many popular tags. We introduce *MultiAspectForensics*, a novel tool to automatically detect and visualize bursts of specific subgraph patterns within a local community of nodes as anomalies in a k,k-hypergraph, leveraging scalable tensor analysis methods. One such pattern consists of a set of vertices that form a dense bipartite graph, whose edges share exactly the same set of attributes. We present empirical results of the proposed method in detecting synthesized anomalies added to some real data, and discuss insights derived from patterns discovered on three datasets from distinct application domains. Moreover, we empirically show that our algorithm can be feasibly applied to higher dimensional datasets. This work was published in [110], and refined version was published in [111].

4.1 Introduction

Modern applications in the Internet era, either data-informed or data-driven, have contributed to the boom of network data arising from a spectrum of domains, such as web knowledge bases [17], network traffic monitoring [62] and online social networks [21]. A glowing trend in the accumulation and analysis of such data is the emergence of heterogeneous interactions between nodes in the network, for which a vivid depiction is offered by the Facebook friendship page, with multiple page elements ranging from wall posts, comments, and photos, to mutual friends, shared interests and common networks between a pair of users. In web knowledge bases, the Resource Description Framework (RDF) is a method for expressing knowledge as *triples* in the form of *subject-predicate-object* expressions that represents a heterogeneous interactions between *subject* nodes and *object* nodes with *predicate* edges [92], and OWL is a language based on RDF that describes the semantics of ontology [15]. The RDF-based knowledge is published as Linked Data [16]. Browsing and navigation over such a space of information, despite its overwhelming scale and complexity, has been a challenging task commonly encountered in many fields. Yet the rather recent availability and popularity of these data, in addition to practical requirements over the efficiency, robustness and generalizability of the solution, has rendered the topic of pattern mining for heterogeneous network data a relatively under-explored one, where even the definition of interesting or abnormal *patterns* could become a non-trivial problem itself.

Many of pioneering studies on pattern discovery for graph and network data focused on frequent substructure mining, with heuristics motivated by information theory [38], mathematical graph theory [169, 99], inductive logic programming [44], etc. An intimately related problem is the detection of rare event and anomalous behavior, which has attracted wide interests thanks to its many well-recognized applications concerned with security, risk assessment, and fraud analysis. Noble and Cook [126] were among the first to address this challenge on structured network data by providing solutions based on the minimal description length principle to search for abnormal subgraphs. And many alternative approaches are now available to spot anomalous nodes [7], edges [30], or both [50], with further elaboration adapted to bipartite graphs [148], and time-evolving graphs [157]. This piece

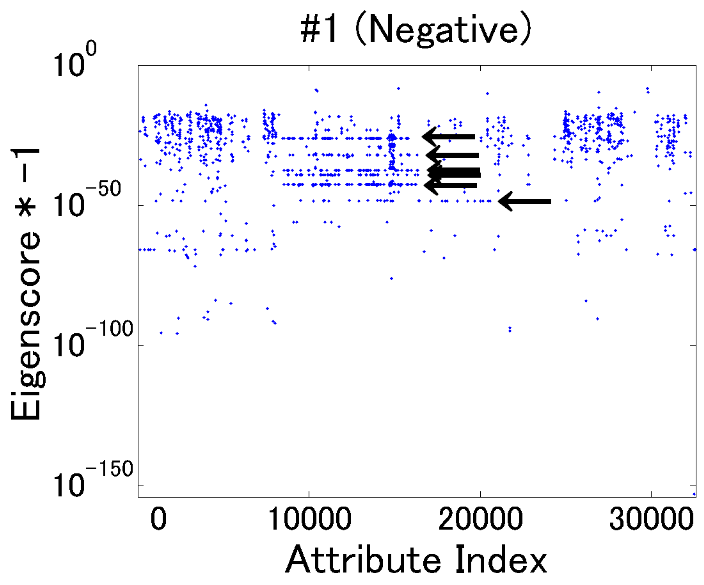


Figure 4.1: A plot which displays absolute values of eigenscores (y-axis in log-scale) along its elements (indexed by the x-axis) for the "IP-source" mode with negative eigenscores for LBNL traffic logs (which we call "*attribute plot*"). Elements are sorted such that IP-sources located in the same local network have similar attribute index. The black arrows point to common score values, illustrating an observation critical to the algorithmic design of *MultiAspectForensics*.

of work, by revealing two classes of patterns in the context of heterogeneous graphs, resembles a novel attempt to explore this relatively young realm of multi-aspect network data for state-of-the-art discoveries and developments.

A heterogeneous network can be represented as a graph, or a k,k-hypergraph, with several statistics that differ according to the types of the relationships between nodes, and these statistics are carefully mixed up to process some data mining tasks such as clustering [151] or classification [80]. Suspicious behaviors such as spam bookmark and distributed cyber attack tend to form clique-like patterns of k,k-hypergraph, in which hyperedges exist between almost every set of k nodes in different partitions. An important and non-trivial problem is to detect such clique-like patterns in a k,k-hypergraph with millions of edges in a scalable way, and distinguish them from those caused by NOT suspicious behaviors, such as many popular contents bookmarked by many users with many popular tags. We resort to a tensor-based representation of a k,k-hypergraph and employ off-the-shelf decomposition algorithms [97] as a starting point of the analysis. Previous research along this line has paid a great deal of attention on individual nodes, which play a central role in similarity ranking [54], personalized recommendation [173], etc. The major finding in our study is that, for multiple datasets across diverse application domains, we could sometimes observe groups of elements with similar connections along one or more data modes, as implied by nearly-identical decomposition scores. Figure 4.1 is an example of the decomposition scores for the "IP-source" of a network traffic logs , which displays absolute values of eigenscores (y-axis) along its elements (indexed by the x-axis). The y-axis should be in *log* scale, because it is expected that nodes of different community structures are separated in *log* scale as discussed in Section 3.2. The black arrows indicate score values shared by many elements, which are not uncharacteristic in other dimensions and across different datasets. *This key observation* enables us to create effective heuristics to extract spikes from histograms as anomalies, and subsequently examine subgraph patterns they imply. And the surprising fact is that many spikes do not appear at the very top of the figure with most significant eigenscore values. While algorithms in aforementioned studies mostly look for elements with top eigenscores, our heuristic distinguishes itself by being able to capture patterns formed by less well-connected nodes in the network, which do not necessarily stand out in the eigenspace and are often ignored by other extant techniques. This enables us to realize that there is an abnormally large community whose nodes are hardly related to the *important* nodes which are well related to many nodes in the graph, *e.g.*, abnormally many spam bookmarks on NOT popular contents with NOT popular tags to attract much attention illegally.

In summary, we propose *MultiAspectForensics*, which starts with a data decomposition step for input heterogeneous networks, features a spike detection heuristic to reveal non-trivial substructure patterns. Our method also includes programs to automatically visualize the detected spikes and summarize the subgraph patterns corresponding to the spikes. The advantage of our method is that we extract anomalous communities in both views of the size of communities and the path capacities between the communities and the *important* nodes, by utilizing even very low absolute values of the eigenvectors of CP decomposition. We demonstrate its effectiveness and efficiency by showing experimental studies on some real datasets added with synthesized anomalies. We also present empirical results of executing *MultiAspectForensics* on three datasets from distinct application scenarios, and investigate the discovered patterns which could be leveraged to suggest suspicious activities from network traffic logs such as port-scanning and denial-of-service attack, extract interesting facts from a web knowledge base such as punk musicians or low-cost airline destinations, and report gene function groups in a developmental biology consistent with established theories. Moreover, we empirically show that our algorithm can be feasibly applied to higher dimensional datasets.

The remainder of this chapter is organized as follows: we first briefly sketch related literatures in Section 4.2, and then elaborate on *MultiAspect-Forensics* procedures step-by-step in Section 4.3. Experimental studies on synthesized anomalies are shown in Section 4.4, and empirical results on three datasets are covered in Section 4.5. Lastly, Section 4.6 concludes the discussion and highlights future directions.

4.2 Related Work

The problem we tackle in this chapter is to detect anomalous community structures which are hardly related to the *important* nodes. As we discussed in Section 2.1, none of existing works on anomaly detection in graph data consider the path capacities between all nodes and communities, *i.e.*, focus on anomalous communities separated by several hops from the *important* nodes.

CP decomposition we use in this chapter is one of the well-known tensor decomposition techniques as we introduced in Section 2.4. We use CP decomposition because the eigenvectors have useful properties for our anomaly detection problem, *i.e.*, the eigenscores of nodes decay exponentially as the nodes are separated from the *important* nodes which have the eigenscores of high absolute values, as we discussed in Section 3.2.

Moreover, our work are related to community detection in k,k-hypergraphs. Several modularity-based community detection in k,k-hypergraphs have been discussed as we described in Section 2.2, but they are not for detection of k,k-hypercliques which we define in Section 4.3. The Boolean Tensor Factorization (see Section 2.4) might be able to detect k,k-hypercliques, but it does not necessarily spot anomalous communities as our method does.

4.3 Problem Definition and Algorithm

In this section, we first define the problem to tackle, then describe the details of algorithm we propose. Definitions of symbols used in this chapter is listed in Table 3.1 and Table 3.2.

4.3.1 Problem Definition

First of all, we formally define k,k-hypergraph and k,k-hyperclique.

Definition 1 (k,k-hypergraph) A k,k-hypergraph is a hypergraph $\mathcal{G} =< V_1 \cup \ldots \cup V_k, H >$, where $V_1, \ldots, V_k$ are disjoint sets of nodes, and $H \subset V_1 \times \ldots \times V_k$ is a set of hyperedges.

Definition 2 (k,k-hyperclique) Given a value ϵ between 0 and 1, a set of nodes $< C_1 \cup \ldots \cup C_k >$ in a hypergraph $\mathcal{G} =< V_1 \cup \ldots \cup V_k, H >$ where $C_1 \subset V_1, \ldots, C_k \subset V_k$ is a k,k-hyperclique if $|(C_1 \times \ldots \times C_k) \cap H| \geq \epsilon |C_1 \times \ldots \times C_k|$.

For example, in a 3,3-hypergraph like Figure 1.1(C), if $C_1 =< u1, u2 >$, $C_2 =< c1, c2 >$, $C_3 =< t1, t2 >$, $(u2, t2, c2) \notin H$, and all other members of $< C_1 \times C_2 \times C_3 >$ are included in H, $|(C_1 \times C_2 \times C_3) \cap H| = 7$ and $|C_1 \times C_2 \times C_3| = 8$. In this case, $< C_1 \cup C_2 \cup C_3 >$ is a 3,3-hyperclique under $\epsilon = 0.8$, because $7 \geq 0.8 \times 8$.

Moreover, we define a l-hops path in k,k-hypergraph again.

Definition 3 (l-hops path in k,k-hypergraph) Given a node pair, a sequence of l hyperedges in which each end hyperedge contains each node of the node pair and every successive hyperedges share at least one common nodes is a l-hops path between the given node pair.

We assume anomalous behaviors cause k,k-hypercliques in which nodes are hardly related to the *important* nodes, *i.e.*, there are few l-hops paths of small l between the nodes in the k,k-hypercliques and the *important* nodes, which have many l-hops paths of small l between many nodes in the graph. Under these definitions and the assumption, our problem is described as:

Problem 1 Given a hypergraph $\mathcal{G} =< V_1 \cup \ldots \cup V_k, H >$, how can we spot k,k-hypercliques in which nodes have few l-hops paths of small l to the *important* nodes which have many l-hops paths of small l to many nodes in the hypergraph, automatically and in a scalable way?

4.3.2 Algorithm Overview

An effective method to tackle the problem should (a) project all nodes in a subspace in which nodes without small hops paths to *important* nodes are located in long distances from the *important* nodes, (b) detect strangely large k,k-hypercliques located even far away from the *important* nodes, and (c) visualize and summarize the detected k,k-hypercliques for further inspection.

We propose *MultiAspectForensics* leveraged by low rank approximation of an adjacency tensor which provides such a subspace described above. *MultiAspectForensics*, in a nutshell, consists of the following steps:

- *Data Decomposition*: Take the input heterogeneous network, *i.e.*, k,k-hypergraph, as a tensor and perform the CP decomposition to obtain a factor vector along each data mode.

- *Spike Detection in Histograms*: Iterate over all data modes to obtain histograms and apply the spike detection algorithm.

- *Visualization*: Create *attribute plot*s and *histogram plot*s with detected spikes highlighted.

- *Substructure Discovery*: Identify the induced subgraph for each spike and summarize patterns discovered.

The above procedure just makes use of the strongest component after data decomposition. If the contribution of the component corresponding to the largest weight is not as large, the latter three steps should be carried out over multiple strongest components in a similar fashion.

The running example in this section is LBNL traffic logs, which comes from snapshot of network traffic logs that consist of packet traces in an enterprise network (LBNL/ICSI Enterprise Tracing Project [2, 129]). We abbreviate them as LBNL. In this section we use a triplet in the logs of (*IP-source*, *IP-destination*, and *port-number*), which could be represented as a 3,3-hypergraph. In this chapter we regard this 3,3-hypergraph as a directed network of machine IP addresses with the only edge attribute "port number" and number of packets as edge weights. We borrowed a subset of this dataset within 1-hour time span in this section.

4.3.3 Data Decomposition

A k,k-hypergraph can be represented as an adjacency tensor $\mathcal{A}$ as discussed in Chapter 3. In terms of a heterogeneous network in data mining research communities, *i.e.*, a graph whose edges have several attributes, it can be transformed into a tensor in which attributes of edges and edge sources and edge destinations become three modes of a tensor. Edge weights naturally stay as element values of the tensor. In another definition of a heterogeneous network in which nodes have different *colors*, node colors could also be incorporated by taking a Cartesian product over two end points of an edge,

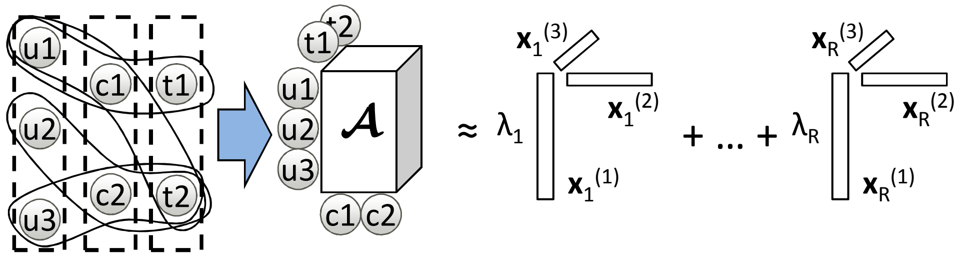


Figure 4.2: Illustration of the CP decomposition: a 3,3-hypergraph is represented as a 3-mode tensor, and decomposed into R triplets of vectors, reminiscing of the rank-R singular value decomposition of a matrix.

for instance, if a directed network contains nodes with 7 different colors, we could have an edge attribute whose arity is $7^2 = 49$.

Tensor decomposition leverages multi-linear algebra to the analysis of high-order data. The CANDECOMP / PARAFAC (CP) decomposition we applied in this chapter generalizes the singular value decomposition (SVD) for matrices. It factorizes a tensor to the weighted sum of rank-one tensor, *i.e.*, the weighted sum of outer products of mode-specific vectors, as illustrated in Figure 4.2 for a 3-order tensor. Formally, for an M-mode tensor $\mathcal{A}$ of size $I_1 \times I_2 \times \cdots \times I_M$, its CP decomposition of rank R yields

$$\mathcal{A} \approx \sum_{r=1}^{R} \lambda_r \left(\mathbf{x}_r^{(1)} \times \ldots \times \mathbf{x}_r^{(M)} \right). \tag{4.1}$$

Similar to SVD, the approximation becomes closer as R enlarges, and would be exact if it equals the rank of the tensor (see [71] for details).

4.3.4 Spike Detection in Histograms

Now that we have transformed complex structured data into a set of more manageable vectors $\mathbf{x}_r^{(m)}$, the next step is to spot *anomalous* patterns from these vectors. Prior to applying the spike detection heuristics, we obtain histogram data which indicate the frequencies of the nodes within the ranges of eigenscores equally divided in log scale. An example of the histogram is shown in Figure 4.3, which we call a *histogram plot*.

Given parameters s, r, and K, the spike detection algorithm (Algorithm 1) just needs to traverse the histogram data until one of the following conditions is satisfied: (1) the energy as measured by the sum of square values of frequencies covered is equal or more than a fraction of s, and the

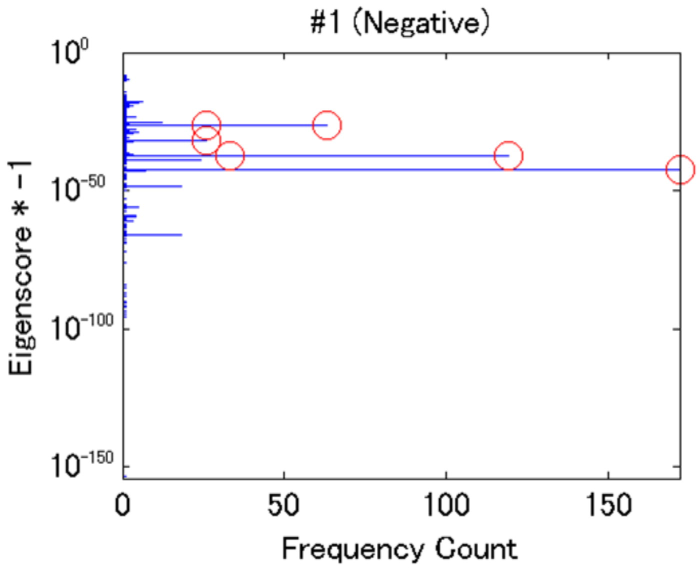


Figure 4.3: A histogram which shows the frequencies of the same nodes of the "IP-source" mode for LBNL traffic logs as Figure 4.1, within the ranges of the eigenscores equally divided in log scale (which we call "*histogram plot*"). The eigenscores are indicated by y-axis, and the frequencies are indicated by x-axis. Detected spikes are indicated by red circles.

frequency is less than a fraction of r of the largest one; (2) K bins are already inspected. After the inspection, the sets of elements within each bin are extracted as spikes, as long as the energy covered is equal or more than a fraction of s. K is the desired number of spikes to be detected. Now we do not have a clear strategy of the parameter settings of s, r, and K, and there is a big room for improvement of our method. In this section and experiments in Section 4.5, we use $K = 20$ as the number of spikes we can investigate practically. We use $r = 50\%$, because the bins with equal or more than half of the frequencies of the largest one should be extracted as spikes, as long as the condition of the energy is satisfied. We have examined several number of s under $K = 20$ and $r = 50\%$, and we chose $s = 90\%$ so that many spikes are extracted for most of the datasets we use. But we use another settings in Section 4.4 so as to pick up synthesized anomalies effectively.

Algorithm 1 *SDA* (Spike Detection Algorithm)

```
Input: Eigenscore histogram vector $H_o$ of size $N$
Output: The set indicating spikes detected $S$
 1: sort the histogram in descending order s.t. $H_{o_1} \geq H_{o_2} \geq \cdots \geq H_{o_N}$
 2: $S \leftarrow \phi$; $Q \leftarrow 0$; $Q_{SUM} \leftarrow \sum_{n=1}^{N} H_{o_n}^2$
 3: for $k = 1, \ldots, K$ do
 4:     $S \leftarrow S \cup \{o_k\}$
 5:     $Q \leftarrow Q + H_{o_k}^2$
 6:     if $Q/Q_{SUM} \geq s$ and $H_{o_k}/H_{o_1} < r$ then
 7:         break
 8:     end if
 9: end for
10: if $Q/Q_{SUM} < s$ then
11:     $S \leftarrow \phi$
12: end if
13: return $S$
```

4.3.5 Visualization

For further inspection of the detected spikes, the distributions of the attribute index of the elements within each spike can be very useful information. For example, if the elements are sorted such that IP-sources located in the same local network have similar attribute index, and the elements within a spike are distributed in a specific range of the indices, we can easily know that these IP-sources are located in the same local network. To investigate such distributions of the attribute index, we create *attribute plot* which displays absolute values of eigenscores by the y-axis in log-scale along the attribute indices by the x-axis as shown in Figure 4.1, and put *attribute plot* on the left side-by-side with *histogram plot* on the right, highlighting every spike in red (Figure 4.4).

The collection of output plots of *MultiAspectForensics* is named *MAF-ray* (*MultiAspectForensics* X-ray), which puts together *attribute plot*s and *histogram plot*s of both positive and negative eigenscores. The rank 1 *MAF-ray* of LBNL traffic logs is shown in Figure 4.5. With *MAF-ray*, users can easily realize that there have occurred spikes in *histogram plot* of some modes, whose corresponding *elements* have specific attribute index in *attribute plot*, like aforementioned spikes of IP-sources.

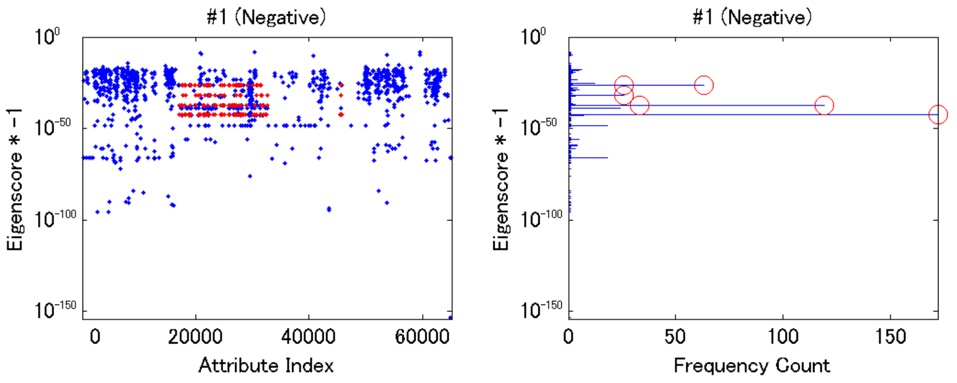


Figure 4.4: An *attribute plot* (adopted from Figure 4.1) on the left side-by-side with the corresponding *histogram plot* (adopted from Figure 4.3). Detected spikes are indicated by red circles in *histogram plot* and red dots in *attribute plot*. We can find the elements within each spike have attribute index in a specific range, indicating that they are located in the same local network.

4.3.6 Substructure Discovery

Having extracted *anomalous* sets of elements that form histogram spikes from each data mode, we head back to the input network data to examine corresponding local subnetworks to complete the final step of pattern discovery. Because the elements within the same spike are expected to behave similarly and specifically, the local patterns corresponding to detected spikes can be understood as bursts of common patterns shared by the elements. As the starting point for analyzing the local patterns, we propose *spike table* that shows detected spikes along with frequency counts of elements within each spike, numbers of common patterns, and numbers of unique elements of other modes within the common patterns. A *spike table* in "IP-source" mode (#1) in our running example is shown in Table 4.1. For example, most of the 119 IP-sources within spike #2 have 3 common patterns (131.243.143.66/534, 131.243.141.187/534, and 131.243.140.105/534 as IP-destination/port number), with 3 distinct IP-destinations (mode #2), and one distinct port number (mode #3). Patterns derived from *MultiAspect-Forensics* can be summarized into the following two categories:

generalized star (*g-Star*)

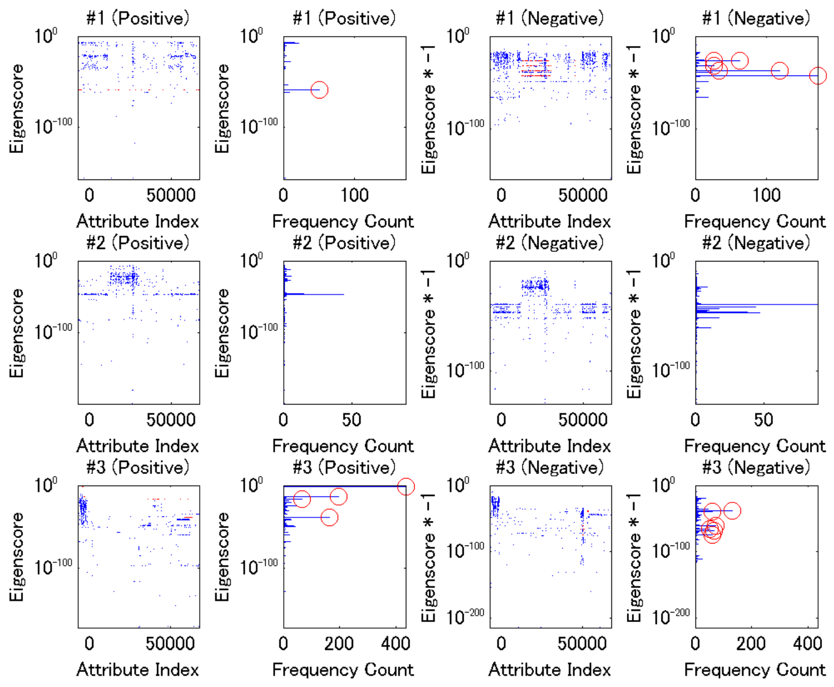


Figure 4.5: 1st *MAF-ray* for LBNL traffic logs: *attribute plot*s and *histogram plot*s, for "IP-source" mode (the top row, #1) and "IP-destination" mode (the middle row, #2) and "port#" mode (the bottom row, #3), and for the positive (left) and negative (right) parts. Using *MAF-ray*, we can quickly spot IP-sources (or IP-destinations, or port#), which have similar behavior, forming one of the patterns in Section 4.3.6.

This pattern can be detected as a spike in a mode, whose corresponding elements share a single common pattern of other modes. This is a subnetwork which consists of conterminous edges that differ only in one data mode. It generalizes the star pattern in bipartite graphs, and makes up a continuous block along one dimension in the adjacency tensor, if elements along that dimension are ordered carefully. We can find 3 spikes of this category in the *spike table* shown in Table 4.1. They are groups of IP-sources sending packets to a single destination server using the same port (Figure 4.6 (a) 'NCP'). Note that in a heterogeneous network, this category of patterns also includes multiple edges between one pair of nodes with differing attribute

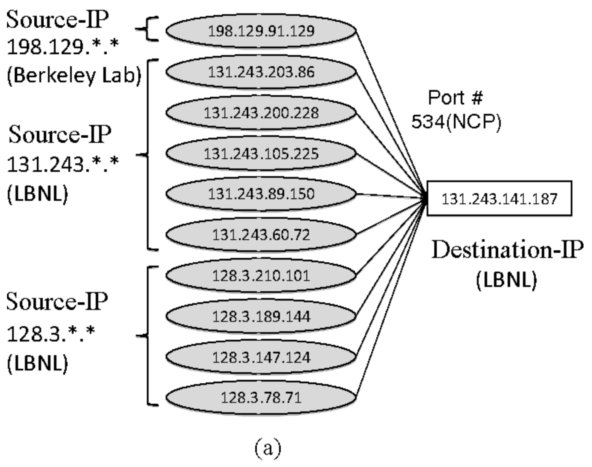
#	**Frequency count**	**# common patterns**	**# unique elements of mode #2;3**	**Description**
1	172	1	1; 1	*g-Star* ('NCP')
2	119	3	3; 1	*g-Bcore* ('File Sharing')
3	63	5	1; 5	*g-Bcore* ('Multi Purpose')
4	51	1	1; 1	*g-Star*
5	33	4	4; 1	*g-Bcore*
6	26	1	1; 1	*g-Star*
7	26	2	1; 2	*g-Bcore*

Table 4.1: A *spike table* in "IP-source" mode (#1) for the LBNL traffic logs. Number of common patterns are the number of patterns of other modes shared by more than 90% of the elements within each spike. Numbers of unique elements of other modes within the common patterns are also shown.

values, *e.g.*, a good many port numbers in our running example, in which case the source machine may be either an administrator performing port screening or a suspect trying to exploit a vulnerable port (Figure 4.6 (b) 'Port Scanning').

generalized bipartite-core (*g-Bcore*)

This pattern can be detected as a spike in a mode, whose corresponding elements share multiple common patterns of other modes. This is a subnetwork that represents a dense bipartite structure similar to the bipartite-core pattern in bipartite graphs. More generally, it can be viewed as a *'slice'* of continuous blocks along two dimensions in higher-order tensors under specific element orders. Patterns of this category can be classified into some classes according to the number of unique elements of each mode within the common patterns. In the *spike table* shown in Table 4.1, spikes of this category are classified into two classes, spikes whose common patterns contain multiple unique elements only in mode #2 (spike #2 and #5), and spikes only in mode #3 (spike #3 and #7). The former class is a group of IP-sources sending packets to multiple destination servers with the same port (Figure 4.7 (a) 'File Sharing'). And the latter class is a group of IP-sources sending packets over different port numbers to the same server (Figure 4.7 (b) 'Multi Purpose'), likely to happen during a DDoS (Distributed Denial-of-



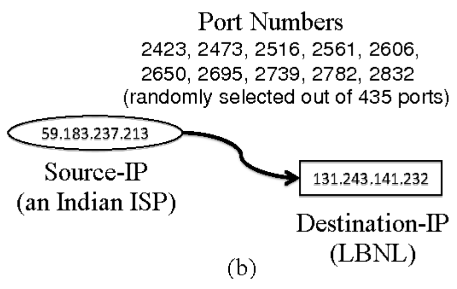
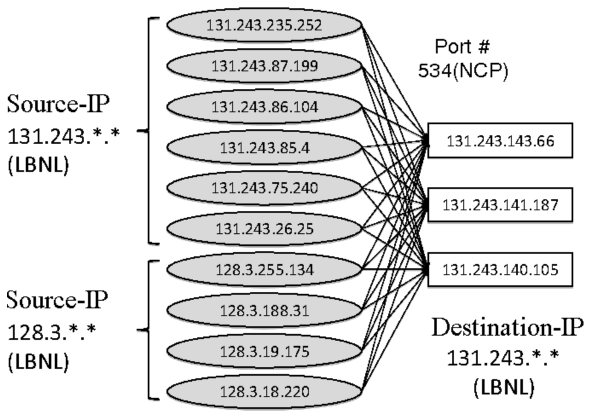
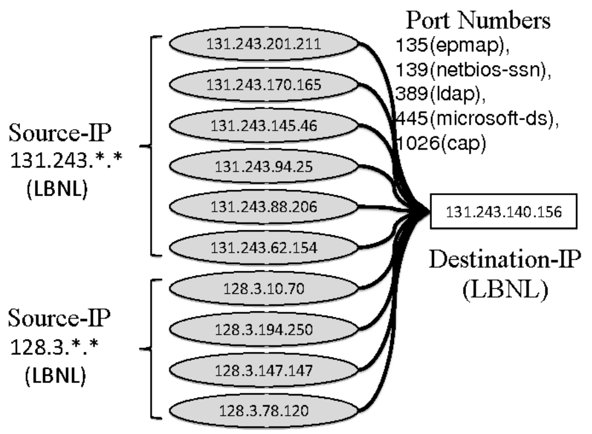


Figure 4.6: Examples of generalized star patterns discovered in the LBNL traffic logs. Wavy arrows indicate multiple edges between the pair of nodes with a handful of distinct attribute values. (a) 'NCP': 10 IP-sources (randomly selected out of 172 ones) are sending multiple packets to a server machine with Port# 524, which is a UDP port under the NCP protocol from a network OS for file sharing and printing services; (b) 'Port Scanning': The IP-source registered by an Indian ISP is sending packets to a host in LBNL via port numbers (ranging from 2,300 to 2,900) not usually intended for this type of communication, implying a suspicious activity.

Service) attack, a typical scenario of network intrusion, in which IP-sources play the role of malicious hosts sending huge volumes of packets to the target server as the victim. Note that this category of patterns can also include classes of spikes whose common patterns contain multiple unique elements in several modes, like spikes detected in the network traffic dataset with additional mode of time tick, as shown in Section 4.5.



(a)



(b)

Figure 4.7: Examples of generalized bipartite-core patterns discovered in the LBNL traffic logs. Wavy arrows indicate multiple edges between the pair of nodes with a handful of distinct attribute values. (a) 'File Sharing': 10 IP-sources (randomly selected out of 119 ones) are sending multiple packets to an array of server machines over a port used for file sharing and printing services; (b) 'Multi Purpose': 10 IP-sources (randomly selected out of 63 ones) are sending packets over different ports to a multi-purpose server machine.

As a final remark, the statement that both patterns are belated to a block along one dimensions or a bundle of blocks in the high-order tensor only holds when elements of their respective data modes are ordered in specific ways. And the complexity to search for such an order is generally exponential, which reflects, in some sense, the power of the proposed approach.

4.4 Evaluation of Accuracy and Scalability

In this section, we present experimental results on the accuracy and scalability of our method. First, we evaluate the accuracy of our method by using 10 largest LBNL traffic logs added with synthesized anomalies. Then we evaluate the scalability by using many LBNL traffic logs of various numbers of packets. *MultiAspectForensics* is implemented in the MATLAB language, and all following experiments are performed on a Unix machine with four 2.8GHz cores, and 16GB memories. We use the *cp_als* function in the MATLAB Tensor Toolbox [11] which features the alternating least squares (ALS) method, a predominant implementation for CP decomposition.

4.4.1 Putting Synthesized Anomalies on Datasets

We create some synthesized anomalies of k,k-hypercliques and add into 10 largest LBNL traffic logs, and evaluate how effectively our method can spot these anomalies. To create k,k-hypercliques located far away from the *important* nodes, *i.e.*, without small hops paths to the *important* nodes which should have the eigenscores of large absolute values, we randomly select elements within the synthesized k,k-hypercliques. These LBNL traffic logs have about $15,000$ to $50,000$ hyperedges, with $1,400$ to $4,500$ nodes of srcIPs, $1,400$ to $4,800$ nodes of dstIPs and $5,400$ to $24,000$ nodes of ports.

Given parameters of *volume* V and *density* ϵ, *i.e.*, a parameter of k,k-hypercliques, we create N anomalies as follows: (1) For each group, we randomly select three values s,d,p between 0 and 1, and decide the number of srcIPs and dstIPs and ports in accordance with the ratio of three selected values, so that sdp is not lower than V, *e.g.*, the number of srcIPs was $\lceil s(V/(sdp))^{1/3} \rceil$ where $\lceil \cdot \rceil$ is the ceiling function. (2) $\lceil V\epsilon \rceil$ elements are randomly selected for each group, that is, hyperedges of a k,k-hyperclique. We tested for $V = 100, 200, 300, 400, 500$, $\epsilon = 1.0, 0.9, 0.8$, and $N = 10$.

4.4.2 Compared Method

The most related method is the Boolean Tensor Factorization (BTF) [114]. Given a binary tensor $\mathcal{B}$, BTF try to find a rank-S tensor

$$\mathcal{C} = \sum_{s=1}^{S} \mathbf{c}_s^{(1)} \times ... \times \mathbf{c}_s^{(M)} \tag{4.2}$$

which minimize $\|\mathcal{B} - \mathcal{C}\|$, where $\mathbf{c}_s^{(m)}$ are binary vectors, using Boolean arithmetic, *i.e.*, defining that $1+1 = 1$. Each component $\mathbf{c}_s^{(1)} \times ... \times \mathbf{c}_s^{(M)}$ represents a group of elements in $\mathcal{B}$ with self-similar indices in each mode, like k,k-hypercliques defined in Section 4.3. As we do not know any implementation of sparse coding of BTF, we compare our method with a similar heuristic which we call *DenseSpot*.

DenseSpot first obtains a rank-S tensor $\hat{\mathcal{C}} = \sum_{s=1}^{S} \hat{\mathbf{c}}_s^{(1)} \times \ldots \times \hat{\mathbf{c}}_s^{(M)}$ which minimize $\|\mathcal{B} - \hat{\mathcal{C}}\|$, where $\hat{\mathbf{c}}_s^{(m)}$ are real-value vectors. This is a relaxation problem of BTF without Boolean arithmetic, and we can obtain a solution by conducting CP decomposition. After that, *DenseSpot* checks entries in s-th component of $\hat{\mathcal{C}}$ corresponding to non-zero elements in $\mathcal{B}$ with indices $(i_1, \ldots, i_M)$ and puts 1 on i_m-th element of $\mathbf{c}_s^{(m)}$ if the entries are greater than a threshold h. Finally, *DenseSpot* selects h, which minimizes $\|\mathcal{B} - \mathcal{C}\|$ and returns those $\mathcal{C}$ calculated by the h. We can easily calculate $\|\mathcal{B} - \mathcal{C}\|^2$ as $\|\mathcal{B}\|^2 - 2\langle \mathcal{B}, \mathcal{C} \rangle + \|\mathcal{C}\|^2$. The s-th group of nodes can be created by selecting value 1 entries of $\mathbf{c}_s^{(m)}$ for all $m = 1, \ldots, M$, regarded as a candidate of k,k-hyperclique. We test $S = 10$ so that *DenseSpot* can detect 10 candidates of k,k-hyperclique.

For *MultiAspectForensics*, we consider a group of hyperedges which contain nodes in a detected spike, and regard a set of nodes contained by all hyperedges in the detected group as a candidate of k,k-hyperclique. For spike detection, we use $s = 10\%$, $r = 80\%$, and $K = 10$, so that our method can detect 10 candidates in maximum.

4.4.3 Accuracy of Detecting Anomalies

We apply *MultiAspectForensics* and *DenseSpot* to LBNL traffic logs added with synthesized anomalies, and compare a candidate of k,k-hyperclique detected by each method with each synthesized anomaly of k,k-hyperclique. We conduct chi-square tests of independence, which assess whether hyperedges of a candidate k,k-hypergraph and those of synthesized k,k-hypergraph are independent of each other. In short, given these two groups of hyperedges, we calculate $\chi^2 = n(a(n-e-g+a)-(e-a)(g-a))^2/(e(n-e)g(n-g))$ where n is the total number of hyperedges, a is the number of common hyperedges between two groups, e and g are the numbers of hyperedges of each group. If χ^2 is greater than a value of p-value at 0.01 of the chi-squared distribution for 1 degree of freedom, we conclude that the method successfully has detected the synthesized anomaly of k,k-hyperclique.

Figure 4.8 is the number of k,k-hypercliques successfully detected by each method. *MultiAspectForensics* can detect almost half of synthesized anomalies even though the volume of k,k-hyperclique are as small as 100, whereas *DenseSpot* can hardly detect such small anomalies. Moreover, *MultiAspectForensics* can successfully detect synthesized anomalies for all ϵ we test, whereas *DenseSpot* seems to focus on anomalies of larger ϵ especially in larger volume as 500. These results suggest that *MultiAspectForensics* can detect *anomalous* k,k-hypercliques located in *local* position of graph structure, which *DenseSpot* does not pay attention to.

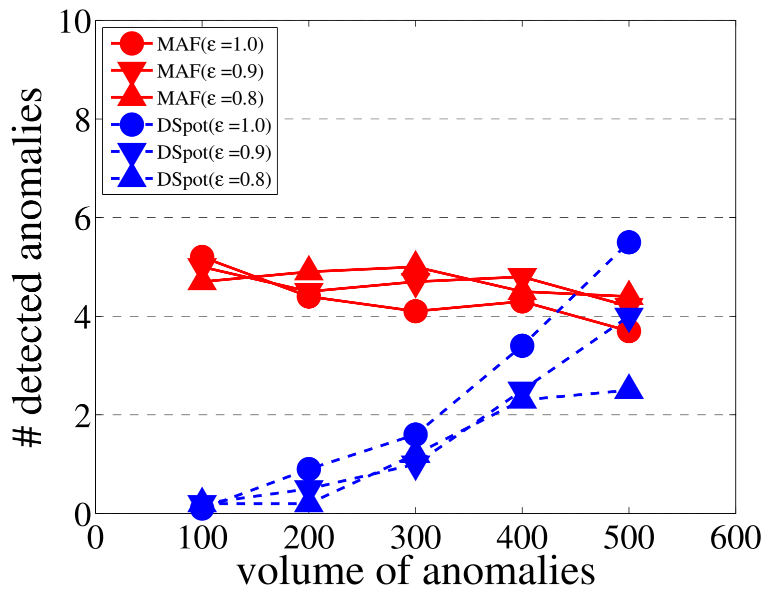


Figure 4.8: Average number of successfully detected synthesized anomalies by each method. Vertical axis: average number of detected synthesized anomalies. Horizontal axis: size of synthesized anomalies. Solid lines: *MultiAspectForensics* (MAF). Dashed lines: *DenseSpot* (DSpot).

4.4.4 Scalability

Can *MultiAspectForensics* be feasibly applied to higher dimensional datasets? The most time-consuming part in *MultiAspectForensics* is CP decomposition for the input tensor, for which a predominant implementation is the alternating least squares (ALS) method, such as the *cp_als* function in the MATLAB Tensor Toolbox [11]. Given a order-M sparse tensor of size $I_1 \times I_2 \times \ldots \times I_M$, the computational cost of each iteration depends on the larger of the following: (1) $I = \sum_{n=1}^{M} I_n$; (2) $NNZ =$ the number of non-zero elements. Figure 4.9 presents empirical results over two different datasets – (a) a traffic log from LBNL for which the tensor is sparse ($NNZ \ll I$), and we add non-zero elements to (a) in order to create a more dense tensor in (b) for which $NNZ \gg I$. These results support the complexity of *MultiAspect-Forensics* in time is $O(N * \max(I, NNZ))$, where N denotes the number of iterations. If the dimension is low and the tensor data stays in the category (b), the computational cost does not depend on the addition of the dimensions, and scales linearly as NNZ grows. When the dimension increases, the tensor data becomes sparser and will fall into the category (a). In such a case, even though the total volume of the high-dimensional space increases exponentially, the computational cost will increase linearly as the dimension grows, proportional with the sum of the number of attribute index. In each case *MultiAspectForensics* has a feasible time complexity for higher dimensional datasets.

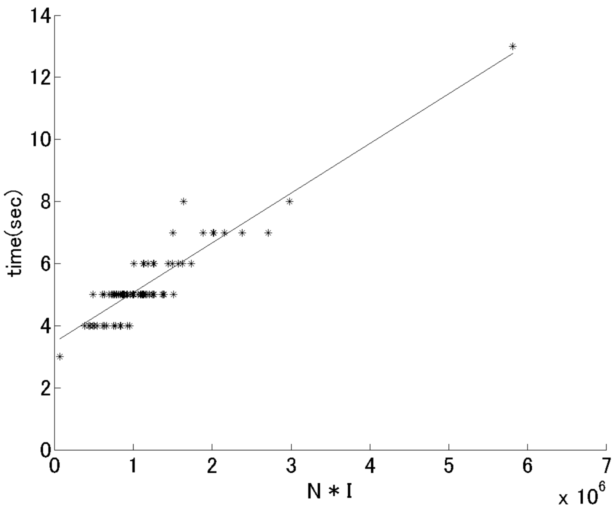
4.5 Empirical Results on Real Data

We discuss about respective patterns discovered by *MultiAspectForensics* in the three datasets.

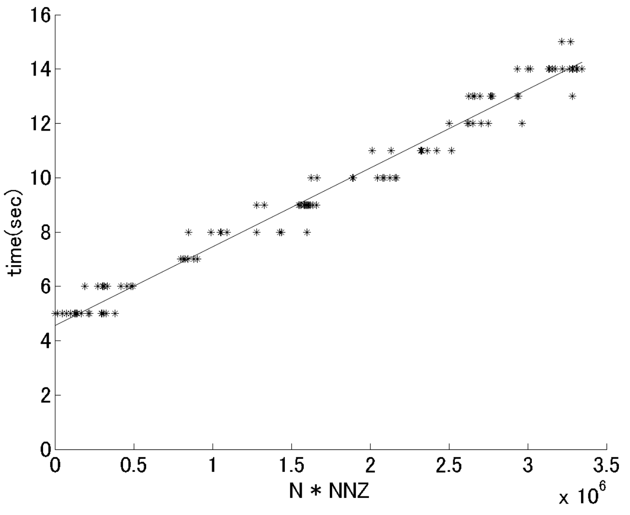
4.5.1 Data and Environment

Datasets are acquired from three dissimilar application domains: network traffic monitoring, knowledge networks, and bioinformatics. A summary is highlighted in Table 4.2. Details of the datasets are described in Section 3.3.

LBNL traffic logs (LBNL) The network traffic log collected on an internal enterprise network of the Lawrence Berkeley National Lab (LBNL) [129]. Each trace in the logs contains *IP-source*, *IP-destination*, *port-number*, and *time tick*. We use a triplet in the logs of (*IP-source*, *IP-destination*, *port-number*) regarded as 3,3-hypergraphs in Section 4.3 and Section 4.4



(a) $NNZ \ll I$



(b) $NNZ \gg I$

Figure 4.9: Computational time of *MultiAspectForensics* on two different datasets. On the x-axis, N denotes the number of iterations, I equals the sum of sizes over all dimensions, and NNZ stands for the number of non-zero elements. The complexity of *MultiAspectForensics* in time is $O(N * \max(I, NNZ))$.

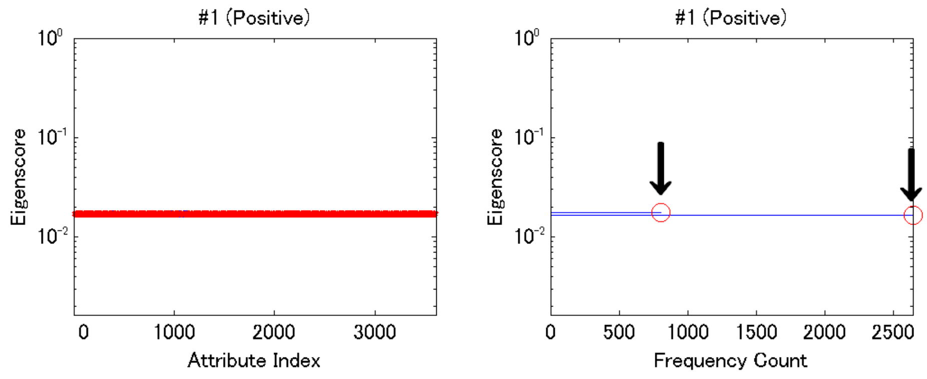


Figure 4.10: An *attribute plot* on the left side-by-side with the corresponding *histogram plot* for the "time" mode (#1) from the 1st *MAF-ray* for LBNL-sdpt. The spikes indicated by black arrow are discussed in Section 4.5.2.

(LBNL). And we use four attributes of (*IP-source*, *IP-destination*, *port-number*, *time tick*) regarded as a 4,4-hypergraph in this section (LBNL-sdpt).

Read the Web (RTW) Online knowledge base derived from the NELL (Never-Ending Language Learning) system at Carnegie Mellon University [27]. Each fact is a triplet of (*subject*, *verb*, *object*) such as (*pittsburgh*, *city-located-in-state*, *pennsylvania*) regarded as a 3,3-hypergraph, which could be also represented as a directed graph made up of entities like *pittsburgh* or *pennsylvania*, edges with attributes like *city-located-in-state*.

Berkeley Drosophila Genome Project (BDGP) A dataset collected from the Berkeley Drosophila Genome Project (BDGP) [153, 154]. We have selected three attributes from the database dump available at [1], which consists of (*genes*, *annotation terms*, *developmental stages*) regarded as a 3,3-hypergraph.

For every of these datasets, the wall-clock time is no more than 2 minutes to carry out the computation and generate *attribute plot* and *histogram plot* along all modes.

Dataset	**# modes**	**Measure**	**# non-zero**	**Dimensions**	**# spikes**
LBNL	3	# packets	27K	2,345 IP-srcs	7
				2,355 IP-dsts	0
				6,055 port #'s	10
LBNL-sdpt	4	# packets	231K	3,610 time ticks	2
				2,345 IP-srcs	0
				2,355 IP-dsts	0
				6,055 port #'s	0
RTW	3	binary	10K	3,641 subjects	15
				98 verbs	0
				3,929 objects	2
BDGP	3	binary	38K	4,491 genes	5
				248 terms	2
				6 stages	0

Table 4.2: A summary of datasets. The numbers of spikes extracted by *MultiAspectForensics* are shown.

4.5.2 LBNL Traffic Log

We have already discussed patterns discovered from a snapshot of this dataset in Section 4.3.6, illustrated in Figures 4.6, 4.7. With the additional mode of time tick (LBNL-sdpt), we have found two dominating spikes for the "time-ticks" mode (Table 4.2). The elements within these two big spikes in *histogram plot* (arrows on Figure 4.10) distributes on almost all the attributes in *attribute plot*, indicating the traffic corresponding to these spikes occurred at almost every time-tick. A *spike table* indicates both of the spikes are *g-Bcore* pattern, bipartite-cores between "time-tick" elements and patterns of other modes (Table 4.3). Upon closer examination, we report the following activities: the first spike is related to the HTTP traffic on port 80 between four servers in LBNL and three remote hosts in Chinese academic institutions, possibly executing scripts to crawl/download web pages. The second spike seems to be related to the same HTTP traffic as the first spike, with additional traffic between a server in LBNL and a remote host at India aforementioned. We have traced further in time and found that the remote host never sent packets back to acknowledge the connection, suggestive of

#	Frequency count	# common patterns	# unique elements of mode #2;3;4	Description
1	2,641	10	9; 9; 7	*g-Bcore*
2	803	11	9; 10; 7	*g-Bcore*

Table 4.3: A *spike table* in "time ticks" mode (#1) for LBNL-sdpt. Number of common patterns are the number of patterns of other modes shared by more than 90% of the elements within each spike. Numbers of unique elements of other modes within the common patterns are also shown.

suspicious activities to be reported to domain experts.

4.5.3 RTW Knowledge Base

Recall that each item in the knowledge database can be represented as a (subject, verb, object) triplet. *MultiAspectForensics* has detected 15 spikes in "subjects" mode and 2 spikes in "objects" mode (Table 4.2). A *spike table* shows the spikes detected in "subjects" mode in Table 4.4. Almost all of the spikes are *g-Star* pattern with exception of a spike of *g-Bcore* pattern having only 2 common patterns.

Figure 4.11 illustrates a subgraph discovered revealing a *g-Star* pattern ('Punk'). The music artists/bands listed here are specialized to punk music according to the knowledge base. And Figure 4.12 displays another *g-Star* pattern ('Ryanair') between European cities and an Irish low-cost airline which flies to many regional or secondary airports to reduce cost, following a different business model and choice of destination from industrial giants.

We should note that some elements within each spike have more versatile peers. For example, some of the musicians of "subjects" mode in the 'Punk' spike have patterns of verb 'music-artist-genre' with objects 'horror_punk', 'proto_punk', 'british_punk' and 'punk_rock' (not shown in the figure). In 'Ryanair' spike, some of the cities of "subjects" mode have patterns of verb 'city-located-in-country' with objects 'finland', 'norway', 'austria', 'scotland', 'spain', 'belgium' and 'ireland' (not shown in the figure). These patterns are not be favorably selected by *MultiAspectForensics*, because they are less important in the first rank.

Moreover, as a sanity check, since node names are ordered alphabetically in this dataset, the pattern does not make a continuous block in the tensor

#	**Frequency count**	**# common patterns**	**# unique elements of mode #2;3**	**Description**
1	265	1	1; 1	*g-Star*
2	134	1	1; 1	*g-Star*
3	63	1	1; 1	*g-Star*
4	31	1	1; 1	*g-Star*
5	30	1	1; 1	*g-Star*
6	30	1	1; 1	*g-Star* ('Punk')
7	29	1	1; 1	*g-Star*
8	27	1	1; 1	*g-Star* ('Ryanair')
9	27	1	1; 1	*g-Star*
10	26	1	1; 1	*g-Star*
11	25	1	1; 1	*g-Star*
12	24	1	1; 1	*g-Star*
13	24	1	1; 1	*g-Star*
14	22	2	2; 2	*g-Bcore*
15	22	1	1; 1	*g-Star*

Table 4.4: A *spike table* in "subjects" mode (#1) for RTW knowledge base. Number of common patterns are the number of patterns of other modes shared by more than 90% of the elements within each spike. Numbers of unique elements of other modes within the common patterns are also shown.

without non-trivial permutation.

4.5.4 BDGP Gene Annotation

In this dataset *MultiAspectForensics* spots a spike of a set of genes known to be responsible for the *maternal effect* in the early development of fruit fly, which also provides hints to study other higher organisms including *Homo sapiens*. Products of such maternal effect genes, in the form of either protein or mRNA, play a critical role in the very early stage of embryo development, such as the first few cell divisions. For instance, four of such genes, including *bicoid*, *caudal*, *hunchback*, and *nanos*, is mostly responsible for the determination of anterior-posterior axis – which side of the embryo will be the future head and which other side will be the future tail [100].

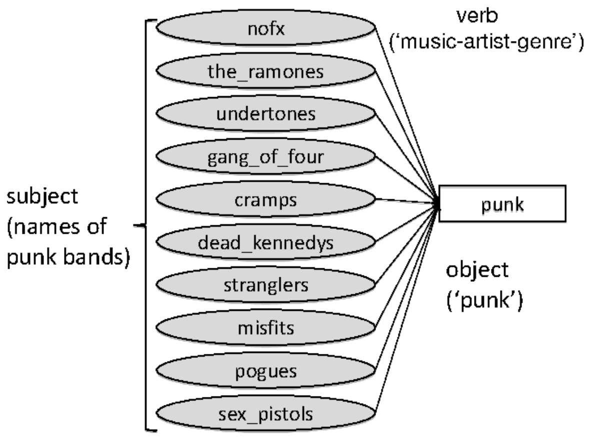


Figure 4.11: 'Punk': A *g-Star* pattern discovered from the RTW knowledge base about 49 punk music artists, of which a random selected set of 10 are listed. They are all specialized in punk or one of its sub-genres according to the knowledge base.

4.6 Conclusion

We presented *MultiAspectForensics*, a novel and effective tool to automatically detect and visualize a category of anomalous patterns in k,k-hypergraphs, including generalized star and generalized bipartite-core patterns. These patterns can be understood as bursts of specific subgraph patterns within a local community of nodes in heterogeneous networks, *i.e.*, k,k-hypercliques described in Section 4.3, even if they exist among less-well connected nodes which are more likely to be ignored by many extant methods. We show our method can effectively detect such patterns by detecting synthesized anomalies added to some real datasets. Empirical results exhibited valuable insights derived from pattern discovered, across multiple application domains such as network traffic monitoring, knowledge networks, and bioinformatics. These successes could be attributed to the fact that we resorted to a tensor-based representation to facilitate data decomposition, reached a key observation leading to spike patterns in histogram plots, and revealed typical substructures reflecting spectral properties of heterogeneous data. Moreover, *MultiAspectForensics* is scalable to higher dimensional datasets, as we have empirically shown.

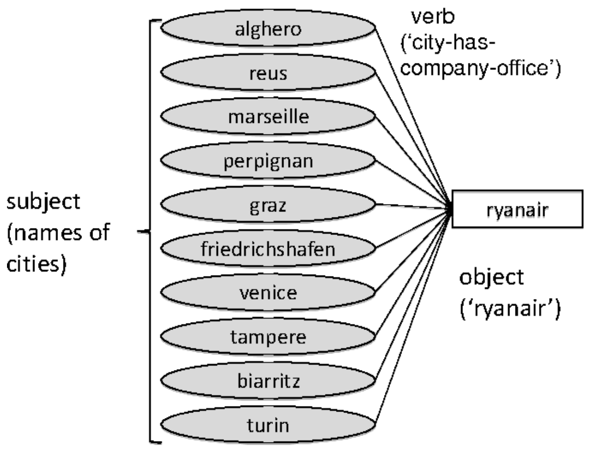


Figure 4.12: 'Ryanair': A *g-Star* pattern discovered from the RTW knowledge base about 36 European destinations of the Ryanair, an Irish low-cost airline, of which a random selected set of 10 are listed. Many of these cities have only sparse connections with other verbs.

Chapter 5

Pattern Discovery Using Singular Value Decomposition

So far we only looked at anomalous community structures themselves in Chapter 4. The more complicated task is to detect anomalies in the relationship of groups of nodes and community structures. To detect such anomalies, we should recognize *patterns* related to community structures, including a near clique or a set of nodes bridging two or more near-cliques or other, and detect anomalies or outliers that deviated from the *patterns.* This would improve intrusion detection in computer networks and network traffic monitoring.

We propose *EigenDiagnostics*, a fast algorithm that spots *patterns* of the nodes related to community structures in a large bipartite graph, such as a pattern of the ratio of the degrees and the numbers of paths to a community, a pattern of the ratio of connections straddling two communities, and so on. The process creates scatter-plots of the node properties (such as singular scores, degree, and weighted degree), then looks for *linear-like patterns.* Our tool automatically discovers such plots, using the Hough transform from machine vision. None of existing works on anomaly detection in graph data consider the path capacities between all nodes and communities, *i.e.*, determine *normal* connection patterns between nodes and communities.

We apply *EigenDiagnostics* on a wide variety of synthetic and real data (LBNL computer traffic, movie-actor data from IMDB, Patent citations, and more). *EigenDiagnostics* finds *patterns*, which appear to correspond to port-scanning (in computer networks), repetitive tasks with bot-net-like behavior, patterns of actors changing careers (in movie-actor data) , and more. The advantages are: (a) it is effective in discovering *patterns* of the nodes related to community structures. (b) it is fast (linear on the number of edges) (c) it is general, and applicable to many, diverse graphs, spanning *tens of GigaBytes*.

This work was published as [109].

5.1 Introduction

Given a large graph, like who-bought-what in a purchasing data or who-contacts-whom in a computer communication network, how can we automatically monitor and report strange behavior of nodes or groups of nodes? To attain such objectives, we should recognize *ordinary patterns* or *laws* among nodes and edges, and detect anomalies or outliers that deviated from them. Recent studies on anomaly detection of graph data provided several solutions for this task [126, 50, 30, 150, 157, 156, 7], though it is more complicated task to detect anomalies in the relationship of groups of nodes and community structures. For example, in a purchasing data, if a group of products which are usually bought by a specific customer segments are bought by some customers who belong to other segments, *e.g.*, cosmetics usually bought by groups of young women are bought by some men in strangely much amount, they are suspicious. In a computer communication network, if a set of servers which are usually accessed in a distributed manner, *e.g.*, a group of mirror web-servers, are accessed at strangely much amount on all of the servers (in *not* a distributed manner), it can be declared to be anomalous access. However, the *normal* connection patterns among a potentially vast amount of community structures in real world data include too many possible types to deal with them comprehensively. As a result, it is a hard task to spot the *anomalous* connection patterns among community structures effectively.

Our intuition is that the singular scores of nodes, *i.e.*, values of corresponding elements of singular vectors of an adjacency matrix, are correlated with the amount of connection with specific community structures as discussed in Section 3.2, and we could comprehensively overlook *patterns* related to many community structures by using small number of singular vectors. A vivid example is the *linear pattern* of the scatter-plot of Figure 5.1, which will be described the details very soon. In a nut-shell, every point corresponds to a network node, and the two axes correspond to two numerical properties, like singular score, degree, weight, etc. By examining some of the nodes on this pattern (red diagonal line), we see that they correspond to the gray nodes of Figure 5.1(b), which clearly straddle the two bipartite cores, which serve as 'bridges' between them.

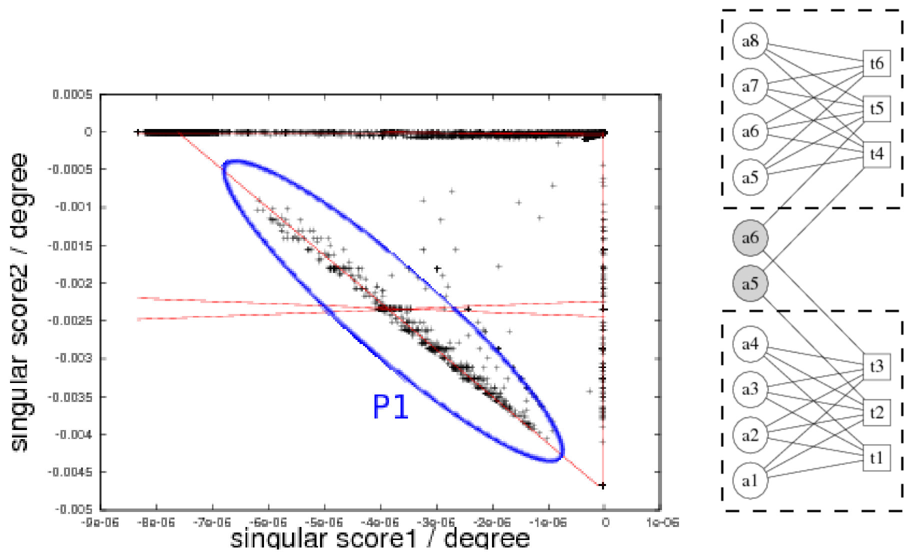
Could we have a method that will automatically do *attention routing*, spotting the nodes that deserve further inspection? Ideally, the method should (a) examine a carefully selected collection of such scatterplots, (b) automatically determine striking patterns, and (c) return some of the network nodes that constitute these patterns. This is exactly what our proposed *EigenDiagnostics* does. In the first part, we propose to use spectral properties of the nodes (score on the first singular vector, 2nd, etc.), as well as traditional measures like degree and total weight. In the second part, automatic detection - we propose to use the *Hough transform* adapted from machine vision, to spot linear constellations of points. In the third part, we propose RepPick, to select representatives from each pattern.

As we show later (Section 5.5), there are several recurring *patterns* in many of the graphs we inspected. One of the most striking cases is the *pattern* of "accidentally related group". These are many nodes that belong to various community structures but *accidentally* connected to a common community structure, such as actors of various genre who *accidentally* co-starred with a few actors who changed careers from adult movies to television shows. Another recurring *pattern* is the *pattern* of "bridging tasks". This is a group of source-IPs which used only two groups of port-numbers, *e.g.*, a group of business use computers which only use two groups of port numbers for sending or receiving e-mails. Such *patterns* themselves might be interesting, while it could be expected to detect anomalous behaviors that deviated from these *patterns*.

We want to emphasize that "accidentally related group" and "bridging tasks" are only two of the many *patterns* we found. Several more *patterns* are described later in Section 5.3, like "division of labor", "satellite group", and many more. All those *patterns* were suggested by *EigenDiagnostics*, and automatically detected.

The main contributions of this work is the design and evaluation of '*EigenDiagnostics*'. This can spot *patterns* of the nodes related to community structures in large graphs by automatically detecting *linear-like patterns* in our proposed SDSD-plot and DS-plot (see Section 5.3). These *pattern* include a pattern of the ratio of the degrees and the numbers of paths to a community, a pattern of the ratio of connections straddling two communities, and so on. Our '*EigenDiagnostics*' method has several desirable properties:

- *Automatic:* it automatically detects *linear-like patterns* (using the



(a) SDSD-plot of 'bridging tasks' in intranet traffic log

(b) template of 'bridging tasks'

Figure 5.1: 'bridging tasks' in a bipartite graph, automatically detected by *EigenDiagnostics*. (a) is a scatter plot of network nodes ("SDSD-plot" - see details in section 5.3); with red lines signifying automatically detected *patterns*; (b) gives the vicinity of selected nodes on the red diagonal line P1 indicated by blue circle in (a) (actually, the template of the pattern) - notice how the black-circle nodes straddle the two bipartite cores (each indicated by a dashed rectangle).

Hough transform), and picks up representative nodes to show to the users ('RepPick' algorithm).

- *Effective:* we have discovered novel and strange *patterns* in multiple real datasets.
- *Scalable:* The method is linear on database size (# of edges).

We run experiments on several diverse and real large bipertite graphs. For weighted graphs, we use LBNL traffic logs and Click Stream. For unweighted graphs, IMDb, DBLP, and Patent Citationare used.

The rest of the chapter is organized as follows: We review the related work in Section 5.2, and proposed strategies are presented in Section 5.3.

We introduce an automated tool in Section 5.4. The experimental results are presented in Section 5.5, and we conclude this work in Section 5.6.

5.2 Related Work

As we discussed in Section 2.1, none of existing works on anomaly detection in graph data considered the path capacities between all nodes and communities, *i.e.*, determined *normal* connection patterns between nodes and communities. Akoglu *et al.* [7] is the most related work which found the *laws* of the connection patterns between each node and "ego-net", *i.e.*, subgraphs of neighboring nodes of each node, and detected abnormal nodes that deviated from the *laws*. However, they did not find the *normal* connection patterns between nodes and community structures.

We use the singular vectors of adjacency matrices which can be used for community detection (see Section 2.2) and measures of *importance* of nodes (see Section 2.3). In addition to these usages of singular vectors, we use them for analysis of the path capacities between nodes and community structures, by utilizing even very low absolute values.

5.3 The Method

In this section, we first define the problem, and then describe our main idea. Most of symbols used in this chapter are listed in Table 3.1 and Table 3.2, and additional symbols are listed in Table 5.1.

5.3.1 Problem Definition

Given a weighted bipartite graph $G =< V_1 \cup V_2, E >$, where $V_1 = \{v_{1i} | 1 \leq i \leq I_1\}$ and $V_2 = \{v_{2i} | 1 \leq i \leq I_2\}$, $E \subset V_1 \times V_2$. The graph G is conceptually stored in a I_1-by-I_2 matrix A, where a_{ij} is the weight of the edge $< i, j >$. The value can be 0/1 for an unweighted graph, or any nonnegative value for a weighted graph. For ease of presentation, we shall refer to the rows of A as 'sources' ('src'), and to the columns of A as 'destinations' ('dst')

Problem 2 How can we detect *patterns* of the number of connections from each node to nodes of several community structures, from the given weighted bipartite (src-dst) graph, automatically and in a scalable way?

5.3.2 Preliminary - Main idea

We propose to use the ratio of singular scores to degrees. More specifically, consider the singular value decomposition (SVD) of the matrix A:

$$A = X\Lambda Y^T \tag{5.1}$$

where Λ is a diagonal matrix with the singular values $\lambda_1, \ldots,$ X and Y contain the left- and right- singular vectors.

Definition 4 (Singular score) The value x_{ri} is the r-th singular score of ('source') node i. Similarly, y_{rj} is the r-th singular score of ('destination') node j.

Symbol	Definition
d_i	weighted degree of i^{th} left node, $\sum_l a_{i,l}$
e_j	weighted degree of j^{th} right node, $\sum_l a_{l,j}$
f_{ri}	x_{ri} divided by d_i, r^{th} left ratio score of A
g_{rj}	y_{rj} divided by e_j, r^{th} right ratio score of A
$\mathbf{1}$	$I_1 \times 1$ vector, all values are 1

Table 5.1: Symbol table (in addition to Table 3.1 and Table 3.2)

The main idea is that, plotting singular scores versus degrees help us discover remarkable *patterns*, as the one in Figure 5.1.

We need one more definition:

Definition 5 (Ratio score) We use ratios of the singular scores (x_{ri} for 'sources', y_{rj} for 'destinations'), divided by the degree of each node. Specifically, for 'source' i we define its r-th left ratio score f_{ri} as

$$f_{ri} = \frac{x_{ri}}{d_i} = \lambda_r^{-1} \frac{\sum_j a_{ij} y_{rj}}{\sum_j a_{ij}} \tag{5.2}$$

and similarly, r-th right ratio score g_{rj} for destination node j:

$$g_{rj} = \frac{y_{rj}}{e_j} = \lambda_r^{-1} \frac{\sum_i a_{ij} x_{ri}}{\sum_i a_{ij}} \tag{5.3}$$

This means that if most of y_{rj} are high for a destination j of high weight a_{ij}, then the score f_{ri} is high, and vice versa. For a unweighted graph, if most of y_{rj} are high for destination j, then the score f_{ri} is high, and vice versa. Left nodes (sources) with dominant r-th left singular scores and right nodes (destinations) with dominant r-th right singular scores form near-clique [135]. Using this fact, the i-th source with high f_{ri} is mainly connected to one or a few near-cliques with a few or many edges. Moreover, nodes in different near-cliques have dominant singular scores of different r, resulting a singular score of different r are correlated with the amount of connections between a node and different near-cliques, as we discussed in Section 3.2.

Based on above intuition, we propose a method '*EigenDiagnostics*', which can rapidly detect *patterns* related to many community structures.

5.3.3 Plotting Properties of the Nodes

We create several kinds of plots to detect *pattern* of f_{ri} (or g_{rj}) among left (or right) nodes. We explain these plots using the results of synthetic data. The synthetic data is very simple unweighted bipartite graph which contains about 40 communities. Nodes in each community are randomly connected with 80 percent of total possible edges (See Table 5.2).

1. **DS-plot**: left (or right) DS-plot (r) is a scatter-plot, with one point for each source node i (or destination node j). It plots the weighted degree d_i (or e_j) vs. the score on the r-th left (or right) singular vector. When r and 'left/right' are omitted, we imply r=1 and 'left' (the main left singular vector). In linear-linear scale, nodes with same ratio score align in a *linear-like pattern*. But *linear-like pattern* in low singular scores cannot be detected in linear-linear scale plot. So we use log-log scale plot in which *linear-like pattern* is observed in *linear-like pattern* with slope 1. If we could observe some *linear-like pattern* with slope of other than 1 in the log-log scale plot, they might be very surprising *patterns*. Nodes with negative singular score are plotted as another DS-plot multiplied by -1.

2. **SDSD-plot**: left (or right) SDSD-plot is a scatter-plot , with one point for each source node i (or destination node j), plotting the first ratio-score f_{1i} (or g_{1j}) versus the second ratio-score f_{2i} (or g_{2j}). When 'left/right' is omitted, we imply left SDSD-plot. The SDSD-plot should be in linear-linear scales.

5.3.4 Patterns in DS-plot

Here we describe some of the *patterns* that we can detect in DS-plot. We also show examples using synthetic data which include new nodes with these patterns. The basic *patterns* we can observe in DS-plot are 'satellite group' pattern in unweighted graph and 'routine task' pattern in weighted graph (Figure 5.2).

Pattern 1 (satellite group) Nodes that are connected to a bipartite core, but not connected to anything else. For example, see the bipartite core in Figure 5.2 (a): nodes a5,a6,a7 are the 'satellites', being connected to the clique with various degrees, but not connected to anything else. Satellites form a sloping line in the DS-plot (red points in Figure 5.2 (a)). The explanation is: Nodes with dominant r-th singular scores form a near-clique [135], in other words, they have high singular scores of same order. So nodes connected only to nodes of a clique have ratio scores of same order, because ratio score is weighted mean of singular scores for the connected nodes. This means these nodes form a sloping line in the DS-plot. See Figure 5.6 (c) for an example from IMDB, where the clique are adult actors (and adult movies), and the satellites are actors who played in only one such movie. More discussion is on Section 5.5.2.

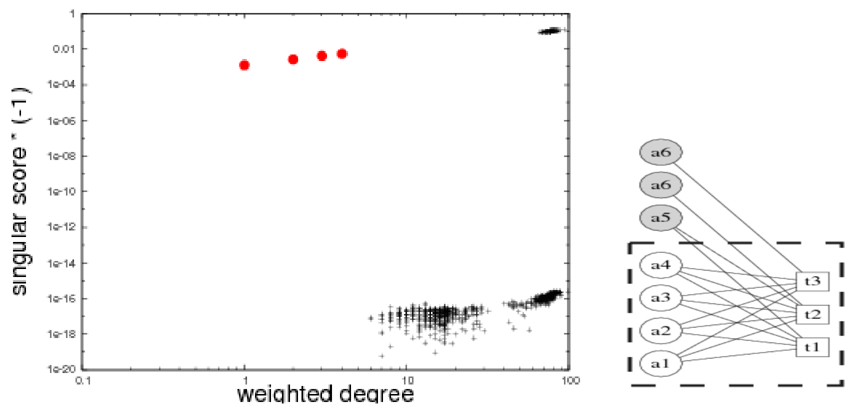
Pattern 2 (routine task) Each node is connected to a specific nodes group with specific weight ratio (Figure 5.2 (b)). Nodes groups with non-zero slopes in DS-plot contains this kind of nodes groups (red points in Figure 5.2 (b)). If some left nodes i have specific weight ratio, their weight vectors $\mathbf{a}_{i:}$ can be written as $p_i\mathbf{b}$, such that $\mathbf{b}$ is a constant row vector. Singular score x_{ri} can be written as $\lambda_r^{-1}\mathbf{a}_{i:}\mathbf{y}_r$, and weighted degree d_i can be written as $\mathbf{a}_{i:}\mathbf{1}$. Thus,

$$f_{ri} = \frac{x_{ri}}{d_i} = \lambda_r^{-1}\frac{\mathbf{b}\mathbf{y}_r}{\mathbf{b}\mathbf{1}} = Const.$$

This means these nodes form non-zero slopes in DS-plot.

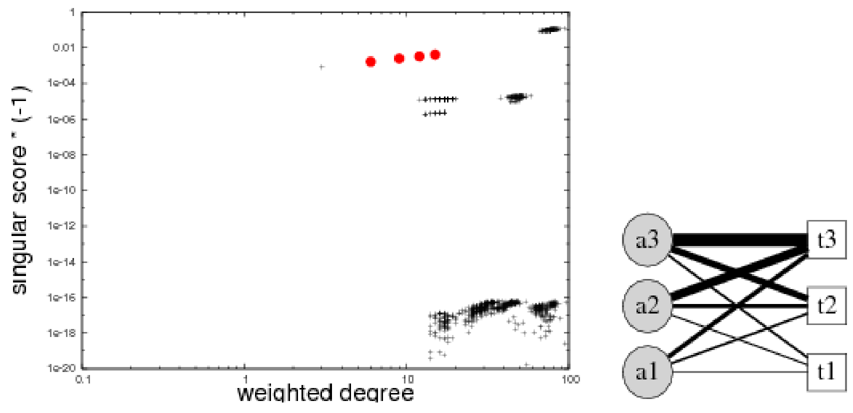
We can observe more remarkable *patterns* in DS-plot such as 'accidentally related group' pattern and 'division of labor' pattern (Figure 5.3). Note that these *patterns* are mainly observed in unweighted graphs, but we can possibly observe them in weighted graphs too.

Pattern 3 (accidentally related group) Nodes connedted to a few nodes which are also connected to nodes within a community, but the connected nodes do not belong to the community. For example, nodes a5, a6, a7 in



(DS-plot)

(a) ’satellite group’

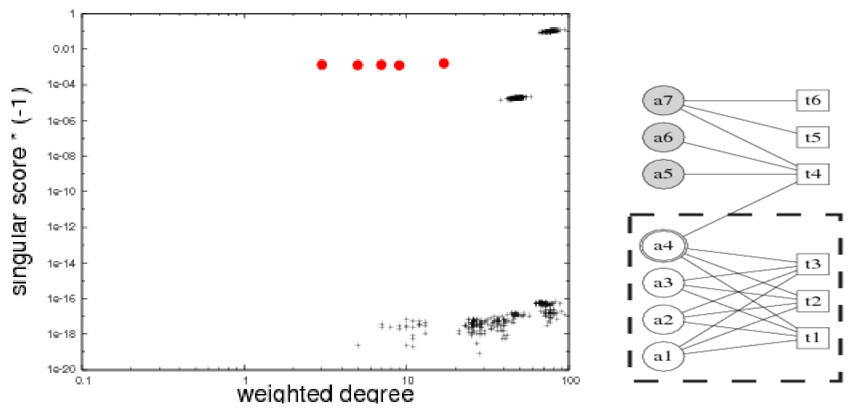


(DS-plot)

(b) ’routine task’

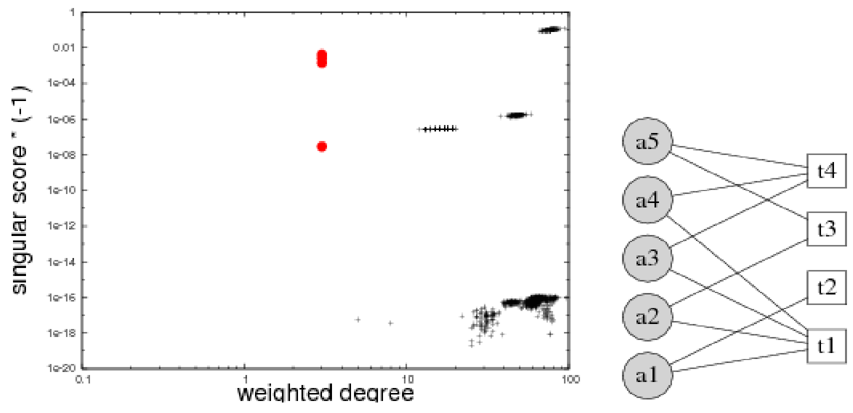
Figure 5.2: Basic *patterns* detected in DS-plot: (a) ’satellite group’: the gray nodes in the template (bottom), act as satellites to the main near-bipartite core. This forms a sloping line in the DS-plot (top row). (b) ’routine task’: the same nodes (in gray) connect to the same destinations (squares), with proportional weights - for the IPsource-port example, that would correspond to a task like checking mail (port 25 for mail, port 993 for imap, as many repetitions as needed). Thickness indicates weight. This (also) shows up as a sloping line in the DS-plot (top row).

Figure 5.3 (a): There is a bipartite core. Node a4 is a member of the bipartite core, but ’accidentally’ connected to outside node t4. The nodes



(DS-plot)

(a)'accidentally related group'



(DS-plot)

(b)'division of labor'

Figure 5.3: More *patterns* detected in DS-plot: (a) 'accidentally related group': the gray nodes 'a5' 'a6' 'a7' in the bottom template, "accidentally" related to the near-bipartite core (indicated by a dashed rectangle). A double circled node 'a4' belongs to the near-bipartite core, but has an "accidental" connection outside it. The gray nodes and the double circled node share the "accidentally" connected node 't4'. They show up as a horizontal line in the DS-plot (red points, in the top diagram). (b) 'division of labor': bottom template shows a sparse bipartite core (like machines scanning ports, after carefully dividing the work. This shows up as a vertical line on the DS-plot (top row).

of accidentally related group(a5, a6, and a7) are connected to the outside node t4 despite they can have other connections. Points that are horizontally distributed in DS-plot correspond to nodes of such node-groups (see red points of Figure 5.3 (a)). The explanation is as follows: Nodes with dominant r-th singular scores form a near-clique [135]. Thus, if a left clique node is "accidentally" connected to an outside right node j, the right node j also has high singular score y_{rj}. As the result, the singular scores x_{ri} for the left nodes i connected to the right node j can be similar to each other. This means these left nodes i are horizontally distributed in DS-plot.

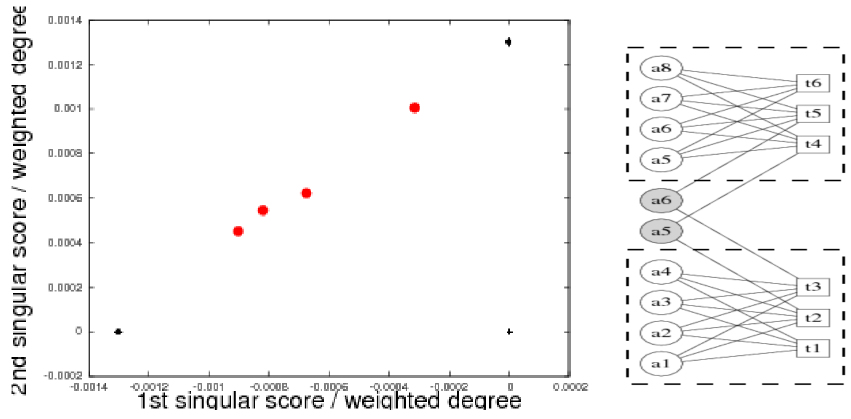
Pattern 4 (division of labor) This pattern consists of nodes that form a large, *sparse*, bipartite core, and all nodes *strangely* have the same degree (see Figure 5.3 (b)). All left side nodes have same degree. Points that are vertically distributed in the DS-plot correspond to this type of nodes (see the red points in Figure 5.3 (b)).

5.3.5 Patterns in SDSD-plot

Next we describe the *patterns* that we can detect in SDSD-plot. Examples using synthetic data which include new nodes with these patterns are also shown . The remarkable *patterns* we can observe in SDSD-plot are 'bridge group' pattern in unweighted graph and 'bridging tasks' pattern in weighted graph (Figure 5.4).

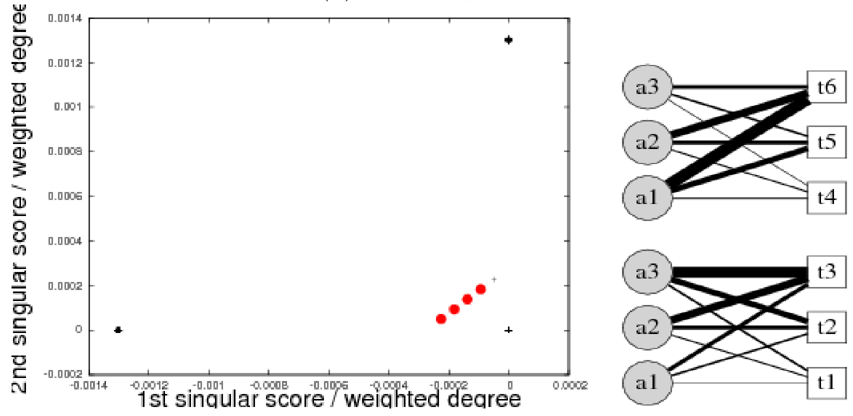
Pattern 5 (bridge group) These are nodes that connect two big communities (see Figure 5.4 (a)). These nodes exhibit a *linear-like pattern* in SDSD-plot, (see red points in Figure 5.4 (a)). The explanation is the following: Suppose that the nodes of the first community have high singular score in their r-th singular vector, and the nodes of the second community have low such scores. However, in s-th singular vector, the situation may be reversed; then, each bridge-node has a score that is linear combination of f_{ri} and f_{si}, and thus such bridge nodes eventually form a line in the SDSD-plot.

Pattern 6 (bridging tasks) Each node is connected to a pair of specific nodes groups with specific weight ratio, but weight ratio differs between two groups (Figure 5.4 (b)). Nodes groups with non-zero slopes in SDSD-plot contains this kind of nodes groups (red points in Figure 5.4 (b)). If some left nodes i are connected to two nodes groups that each group has specific weight ratio, their weight vectors $\mathbf{a}_{i:}$ can be written as $p_i\mathbf{b} + q_i\mathbf{c}$, such that $\mathbf{b}$ and $\mathbf{c}$ are constant row vectors. Left singular score x_{ri} can be written as



(SDSD-plot)

(a) 'bridge group'



(SDSD-plot)

(b) 'bridging tasks'

Figure 5.4: Remarkable *patterns* detected in SDSD-plot: (a) 'bridge group': bottom template shows gray nodes 'a5' and 'a6', straddling the same two (near-) bipartite cores, acting like bridges. This pattern shows up as a sloping line in the SDSD-plot (top row). (b) 'bridging tasks': the same gray nodes, perform *two* groups of tasks (instead of just one, as (b)). This leads to a sloping line in the DS-plot (top row).

$\lambda_r^{-1}\mathbf{a}_{i:}\mathbf{y}_r$, and weighted degree d_i can be written as $\mathbf{a}_{i:}\mathbf{1}$. Thus,

$$f_{ri} = \frac{x_{ri}}{d_i} = \lambda_r^{-1}\frac{p_i\mathbf{b}\mathbf{y}_r + q_i\mathbf{c}\mathbf{y}_r}{p_i\mathbf{b}\mathbf{1} + q_i\mathbf{c}\mathbf{1}}.$$

Similarly,

$$f_{si} = \frac{x_{si}}{d_i} = \lambda_s^{-1} \frac{p_i \mathbf{b}\mathbf{y}_s + q_i \mathbf{c}\mathbf{y}_s}{p_i \mathbf{b}\mathbf{1} + q_i \mathbf{c}\mathbf{1}}.$$

From this, linear combination of f_{ri} and f_{si} can be constant for all i in this group. This means if nodes are plotted at f_{ri} against f_{si}, they forms *linear-like patterns*.

5.4 Proposed Tool: *EigenDiagnostics*

In this section, we introduce '*EigenDiagnostics*', a novel tool to automatically spot groups of nodes with *patterns* related to community structures, in a large sparse graph.

5.4.1 Main Algorithm

The input of '*EigenDiagnostics*' is a weighted bipartite graph, and the output is a set of *pattern*. '*EigenDiagnostics*' reports *patterns* with some representative nodes so that users can easily understand what kind of *pattern* they are. Figure 5.5 is the framework of '*EigenDiagnostics*'. In Figure 5.5: (1) *EigenDiagnostics*' computes some properties of nodes, such as weighted degree and singular score; (2) Nodes are plotted against two properties into 2-D; (3) Lines are automatically detected using Hough transform as *patterns*; (4) Some representative nodes on the detected lines are chosen ('RepPick' algorithm, to be described later), and '*EigenDiagnostics*' reports users the *patterns* with the representative nodes. We give the pseudo-code of '*EigenDiagnostics*' in Algorithm 2.

5.4.2 Modified Hough Transform for Automatic Line Detection

The Hough transform is a feature extraction technique used in image analysis, computer vision, and digital image processing [49]. The Hough transform is a most popular technique to detect lines, circles, etc. '*EigenDiagnostics*' tries to detect *linear-like pattern* using this method. '*EigenDiagnostics*' first converts plot data to an $n \times n$ bitmap, then converts the bitmap data into Hough space, and pick up the peaks in Hough space. The Hough transform accepts as input a 2-d bitmap, and returns 2-d array (ρ, θ), indicating how prominent is the line that has distance ρ from the origin, and orientation θ (the orientation is related to the slope of the line, and it is the angle between the x axis and the vector that is perpendicular to our line). ρ and θ

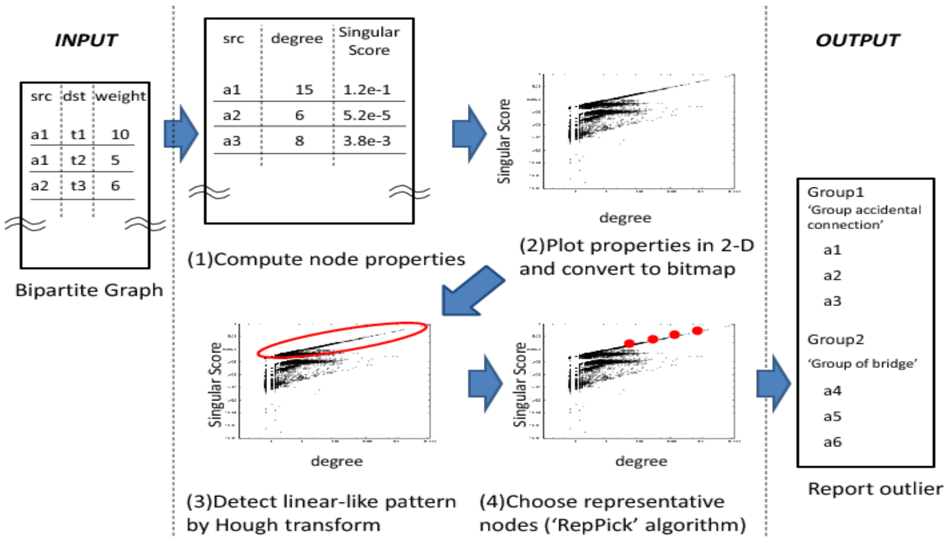

INPUT
src dst weight
a1 t1 10
a1 t2 5
a2 t3 6
Bipartite Graph
src degree Singular Score
a1 15 1.2e-1
a2 6 5.2e-5
a3 8 3.8e-3
(1)Compute node properties
Singular Score
degree
(2)Plot properties in 2-D and convert to bitmap
(3)Detect linear-like pattern by Hough transform
(4)Choose representative nodes ('RepPick' algorithm)
OUTPUT
Group1
'Group accidental connection'
a1
a2
a3
Group2
'Group of bridge'
a4
a5
a6
Report outlier

Figure 5.5: Workflow of *EigenDiagnostics*. Given an input graph, it computes the necessary plots, spots lines (in red), and reports representatives (red circles).

are digitized to user-defined increments. Thus, a horizontal line that passes through the origin should give a high score for the (0,90) entry of the Hough transform, while a vertical line through the origin would lead to highs score at (0,0).

Unfortunately, as-is, the Hough transform is not enough to help us spot prominent lines, because there can be many false-positive lines with high score which pass through some dense areas in the 2-D bitmap. So we want (ρ, θ) entries that (a) have high score and (b) have neighbors with low score. Thus, our method looks for *peaks* in the Hough matrix. The pseudo-code is Algorithm 3.

5.4.3 Pick Up Representative Nodes

On the detected lines are usually too many nodes for users to understand what was happening. So we introduce 'RepPick' algorithm, which can choose representative nodes by which users can check network easily. In short, 'RepPick' chooses nodes at regular intervals on the detected line. Nodes with big values for both properties are preferentially selected. The pseudo-code is Algorithm 4.

Algorithm 2 *EigenDiagnostics*

```
Input: $A$ (the adjacency matrix of a weighted bipartite graph $G =< V_1 \cup V_2, E >$ )
Output: a set of representative nodes of *patterns* $R_t = \{a_{t_1}, a_{t_2}, ...\}$ such that $\forall a_{t_i} \in V_1$, $t = 1, 2, \ldots$
 1: compute each $a_i(\in V_1)$'s properties $p_{k,i}$ such as singular score, weighted degree ,etc. $(1 \leq k \leq K$, $1 \leq i \leq |V_1|)$
 2: $t = 1$
 3: for all combination { $k_1$, $k_2$ }, $1 \leq k_1 \leq K$, $1 \leq k_2 \leq K$, $k_1 \neq k_2$ do
 4:    detect linear-like distributed nodes $O_s = \{a_{s_1}, a_{s_2}, \ldots\}$ in $p_{k_1,i}$ vs $p_{k_2,i}$ plot using Hough transform, $s = 1, 2, \ldots$ (Algorithm 3)
 5:    for all $O_s$ do
 6:       pick up representative nodes as $R_t = \{a_{t_1}, a_{t_2}, \ldots\}$ by 'RepPick' algorithm (Algorithm 4)
 7:       $t = t + 1$
 8:    end for
 9: end for
10: return $R_t$ , $t = 1, 2, \ldots$
11: end function
```

Algorithm 3 detect line using Hough transform

```
Input: node properties { $< p_{k_1,i}, p_{k_2,i} >$ } such that $\forall a_i \in V_1$, $1 \leq k_1 \leq K$, $1 \leq k_2 \leq K$
Output: linear-like distributed nodes $O_s = \{a_{s_1}, a_{s_2}, ...\}$ ,$s = 1, 2, ...$
 1: plot node $a_i$ in 2-D against node properties $p_{k_1,i}$ vs $p_{k_2,i}$
 2: convert plot data to bitmap data
 3: convert bitmap data to Hough space
 4: pick up peaks { $< \rho_s, \theta_s >$ } in Hough space
 5: for all { $< p_{k_1,i}, p_{k_2,i} >$ } do
 6:    if {$< p_{k_1,i}, p_{k_2,i} >$} is near from line { $< \rho_s, \theta_s >$ } then
 7:       put $a_i$ to $O_s$
 8:    end if
 9: end for
10: return $O_s$
11: end function
```

Algorithm 4 'RepPick' algorithm

```
Input: linear-like distributed nodes $O = \{a_{s_1}, a_{s_2}, ...\}$
Input: $p_{k_1,s_i}$, $p_{k_2,s_i}$ such that $\forall a_{s_i} \in O$
Output: representative nodes $R = \{a_{t_1}, a_{t_2}, ...\}$
 1: pick up a node $a_x$ that both $p_{k_1,x}$ and $p_{k_2,x}$ are big absolute values (e.g.,
    a node with max harmonic mean of absolute values)
 2: $maxD$ = max distance from $a_x$ among $a_{s_j} \in O$
 3: for  $h = 0$; $h \leq maxD$; $h = h + Const.$  do
 4:    put a node whose distance from $a_x$ is closest to $h$ into $R$
 5:    if more than $S$ (max representative node count) nodes are selected
       then
 6:       break
 7:    end if
 8: end for
 9: return $R$
10: end function
```

5.5 Experimental Evaluation

5.5.1 Experimental Setup and Datasets

First we describe the experimental setup and the datasets we use. The experiments run on 4 core Xeon 2.8GHz, with Fedora 7 Linux, 16Gb of memory and 2Tb of disk. We use the following datasets. Details are described in Section 3.3.

- IMDb (unweighted): Relationship between about 553,000 authors and 204,000 movies.

- DBLP(unweighted): The author-paper information of DBLP dataset. This data contains about 316,000 authors and 472,000 papers.

- Patent Citation(unweighted): Patent citation data between about 2,089,000 citing patents and 3,259,000 cited patents[67].

- LBNL traffic logs (weighted): The network traffic log is collected on an internal enterprise network of the Lawrence Berkeley National Lab (LBNL) [129]. Each trace in the logs contains *IP-source*, *IP-destination*, *port-number*, and *time tick*. We use two of these information as three bipartite graphs, such as *IP-source* vs *IP-destination*, *IP-source* vs *port-number*, *IP-destination* vs *port-number*. The weight of each graph is $\log(\#ofpacket+1)$. There are two kind of port number, TCP and UDP. And this dataset is separated into two dataset, dataset without scanning activity (we call 'regular' dataset) and dataset with scanning activity (we call 'scanners' dataset). We conduct experiments on combined datasets of all hours for each of the combination of TCP/UDP, regular/scanners. But we show only the results of LBNL regular TCP IP-source vs port-number (LBNL-sp) and LBNL scanners UDP IP-destination vs port-number (LBNL-dp).

- Click Stream(weighted): Click stream data [118]. About 23,000 people clicked link of 199,000 sites. Weight is active time.
- Syn. data(unweighted): Synthetic data. This data contains about 4,000 nodes and 106,000 edges. This data contains about 40 communities. Nodes in each community are randomly connected with 80 percent of total possible edges.

All the above data sets are summarized in table(5.2):

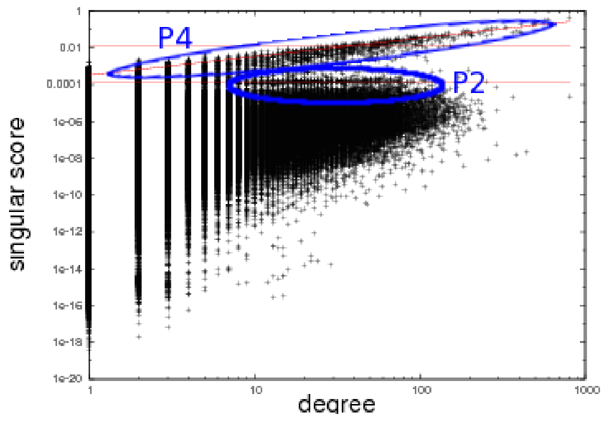
dataset	number of nodes	number of edges
IMDb	$\approx 757K$	$\approx 2,270K$
DBLP	$\approx 788K$	$\approx 1,073K$
Patent Citation	$\approx 5,348K$	$\approx 16,522K$
LBNL-sp	$\approx 66K$	$\approx 3,029K$
LBNL-dp	$\approx 69K$	$\approx 1,339K$
Click Stream	$\approx 222K$	$\approx 953K$
Syn. data	$\approx 4K$	$\approx 106K$

Table 5.2: Summary of data sets

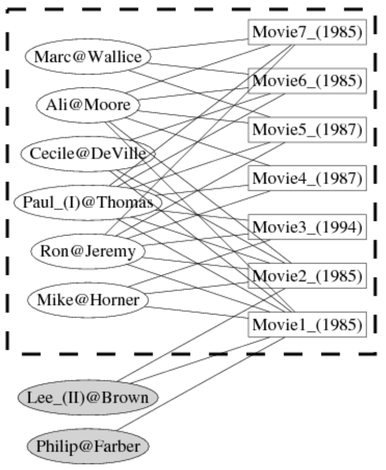
5.5.2 Experiment on Real Data

Due to the page limit, we only show left 1st and 2nd DS-plot and left SDSD-plot for each dataset. The results on unweighted graphs are shown in Figure 5.10, Figure 5.11, and Figure 5.12. We choose the most striking *patterns* among these plots (P2 and P4). We describe these *patterns* as below.

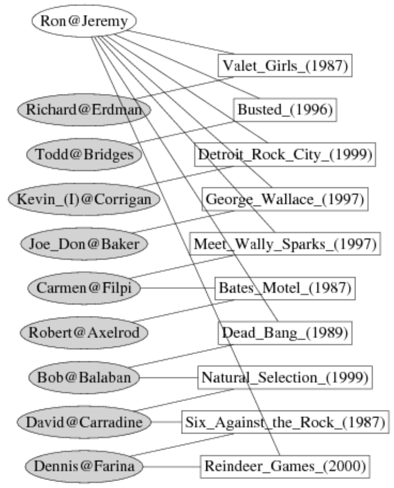
Observation 1 (satellite group) We have found a diagonal line in DS-plot of the IMDb database (P4 in Figure 5.10 and Figure 5.6 (a)). Figure



(a) DS-plot of IMDb data set.



(b) representative actors of P4.



(c) representative actors of P2.

Figure 5.6: Automatic discovery of ‘accidentally related group’ and ’satellite group’: (a) DS-plot of IMDb data set. We have found ’satellite group’(P4) and ’accidentally related group’(P2). (b) representative actors (shaded nodes) which are on the diagonal line of ’satellite group’ (P4) in DS-plot of IMDb dataset. ’Lee(II) Brown’ and ’Philip Farber’ played in a few movies of an adult movie clique (dashed rectangle), and did not play in other movies. (c) representative actors (shaded nodes) which are on the horizontal line of ’accidentally related group’ (P2) in DS-plot of IMDb dataset. ‘Ron Jeremy’ (top oval node) played in many mainstream movies (a few shown, as rectangles), as many adult movies (not shown)

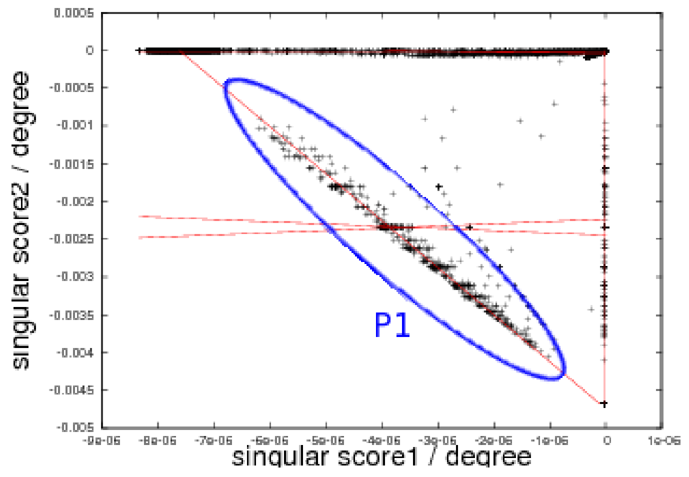
5.6 (b) shows the representative actors on this diagonal line (shaded nodes). There are a near-bipartite core (dashed rectangle) of actors and adult movie films. The representative actors appeared in a few films of the clique and did not appear in other films, like 'satellites' of the clique. We only show the representative actors with low degrees to make the explanation easier. We do not show all the films in which the actors in the dashed rectangle appeared.

Observation 2 (accidentally related group) We have also found a horizontal line in DS-plot of the IMDb database (P2 in Figure 5.10 and Figure 5.6 (a)). There are horizontally distributed actors in DS-plot of 2nd singular scores. Figure 5.6 (c) shows the representative actors on this horizontal line (shaded nodes). They rarely shared movies, but there was an actor who shared movies with almost all of the representative actors. The actor, Ron Jeremy, a star of adult films, has also appeared in many other more mainstream movies. The movies the representative actors shared with Jeremy were not adult movies.

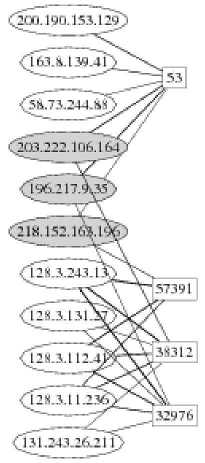
We show the results on weighted graphs in Figure 5.13, Figure 5.14, Figure 5.15. The most striking *patterns* among these plots are chosen (P1, P3, and P5(not shown)). We describe these *patterns* as below.

Observation 3 (bridging tasks) There are *linear-like patterns* which do not contain origin in SDSD-plot of LBNL-dp for f_{1i} and f_{2i} (P1 in Figure 5.14 and Figure 5.7 (a)). This is 'bridging tasks' pattern. We pick up some representative nodes and check what they are. Figure 5.7 (b) shows three representative IP-destinations (shaded nodes) and port numbers they are accessed. We also show other IP-destinations which shared same port numbers with all of the three representative IP-destinations. There is a near-bipartite core which contains port number 57391, 38312, 32976. There is a very big tree in which about 17,000 dst address are accessed to port 53 (DNS). The three representative IP-destinations were accessed by "bridging tasks" on those two groups of port numbers. This means there were two kinds of port scanning, one scans port 53, and another scans over port 30,000 (maybe over 32,768). And there were a group of IP-destinations scanned by both methods.

We have found another 'bridging tasks' pattern in DS-plot of LBNL-sp (P3 in Figure 5.13 and Figure 5.8 (a)). There are many lines in SDSD-plot of ratio scores for the IP-sources. '*EigenDiagnostics*' has detected a striking line which do not contain origin (Figure 5.8(a)). Figure 5.8 (b) shows the



(a) SDSD-plot of LBNL-dp.

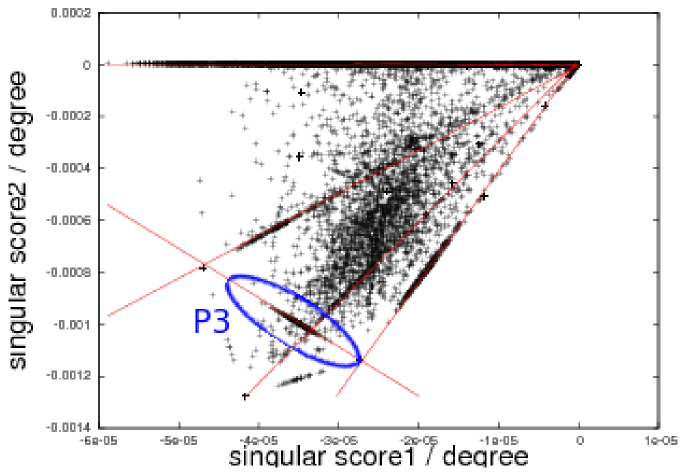


(b) representative IP-destinations.

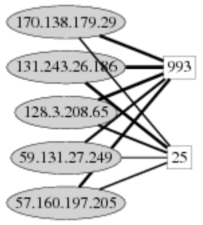
Figure 5.7: An example of a bridging tasks pattern: (a) SDSD-plot of LBNL-dp. (b) representative IP-destination (shaded nodes) and port numbers (right nodes). The shaded nodes used port 53 (dns), as well as high-numbered ports (probably used for P2P traffic).

representative IP-sources on this line (shaded nodes). These IP-sources sent only 25 (SMTP) and 993 (IMAP) which are used to send/receive e-mail. They do not do another operations. They may be computers of secretaries or administrative office. By using '*EigenDiagnostics*', we can easily know there are this kind of computers in intranet.

Observation 4 (division of labor) We have found a group of vertically distributed port numbers in DS-plot for 2nd right singular vector of the LBNL-dp (data not shown). This is 'division of labor' pattern. This pattern means there are many nodes with *strangely* same degrees. In this case, there are several IP-destinations which were connected to various port numbers of this group. Some computers were connected to a few port numbers, whereas other computers were connected to many port numbers. But the number of destination computers to which each port number was connected is *strangely* equal. This may have been caused by the same scanning program, running on a computer or distributed computers.



(a) SDSD-plot of LBNL-sp.



(b) representative IP-sources.

Figure 5.8: Another example of 'bridging tasks' pattern: (a) SDSD-plot of LBNL-sp. There are some 'bridging tasks' patterns. (b) representative IP-sources (shaded nodes) and port numbers (right nodes). The shades nodes used only port 25 (smtp) and 993 (imap).

5.5.3 Scalability

To evaluate the performance of '*EigenDiagnostics*', we report the computation time as we vary the number of graph edges, for various synthetic and real data (Figure 5.9). As expected, it clearly shows that the processing time is linear in the number of graph edges. This is mainly due to the computation time of singular value decomposition (*svds* of MATLAB implementation), which dominates that of *Hough Transform* in our experimental settings.

5.6 Conclusions

The main contribution is the design of *EigenDiagnostics*, a carefully-designed method to spot patterns on large graphs. *EigenDiagnostics* has the following desirable properties:

1. *Automatic:* it operates without human intervention, and does *attention routing*, by giving the user a few nodes that are worth inspecting (like bridge-nodes, members of a bot-net-like group, etc).

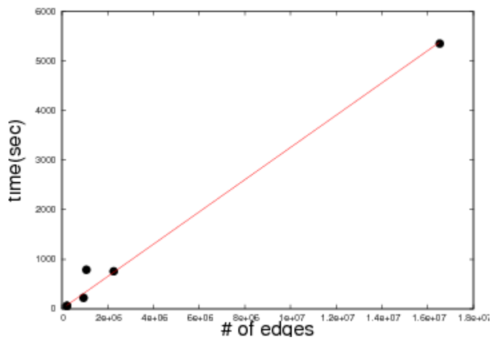
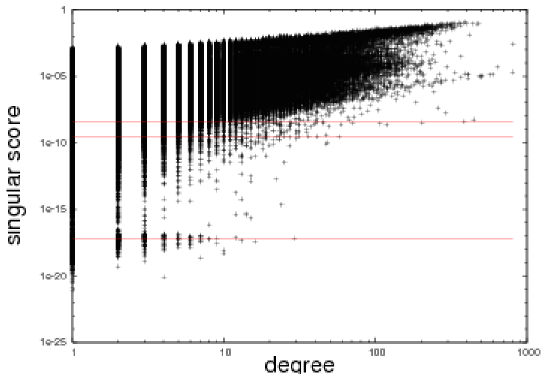


Figure 5.9: Computation time of '*EigenDiagnostics*', versus # of edges. The scaling is linear.

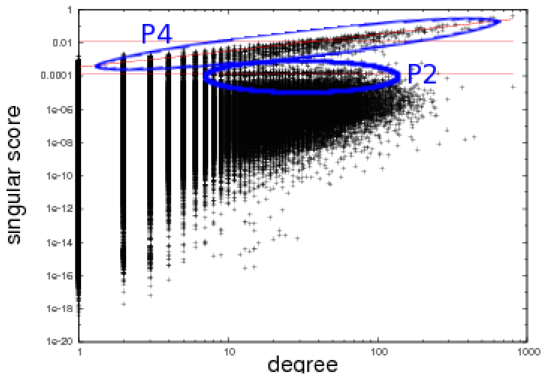
2. *Effective:* We showed that *EigenDiagnostics* discovered novel patterns in multiple real datasets, such as 'satellite group', 'accidentally related group' and 'bridging tasks'. Not only on computer traffic data, but also movie-actor data, and patent citations, etc.

3. *Scalable:* the method is carefully designed, with singular value decomposition being the most expensive operation. Using modern sparse-matrix eigensolvers, the complexity is linear on the database size (ie., linear on the number of edges).

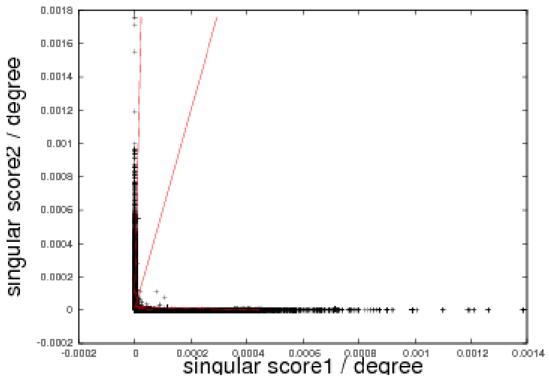
Finally, as our goal is anomaly detection, some additional methods to detect anomalies or outliers based on *patterns* spotted by our method might be required. There are many possible solutions. One possibility is to detect outlier nodes directly on DS-plot and SDSD-plot. Another possibility is to extract groups of nodes located near *linear-like patterns*, and to detect anomalous nodes which deviate from *laws* of number of degrees and edges related to the groups, like [7].



(DS-plot of 1st singular score)

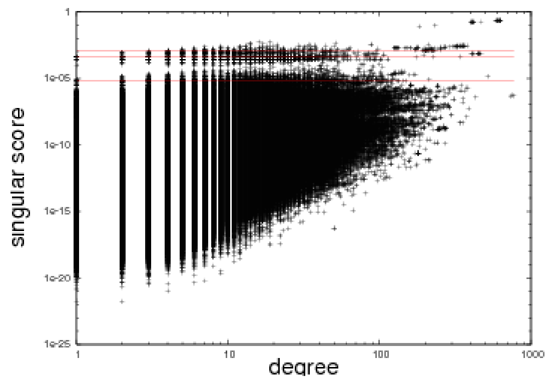


(DS-plot of 2nd singular score)

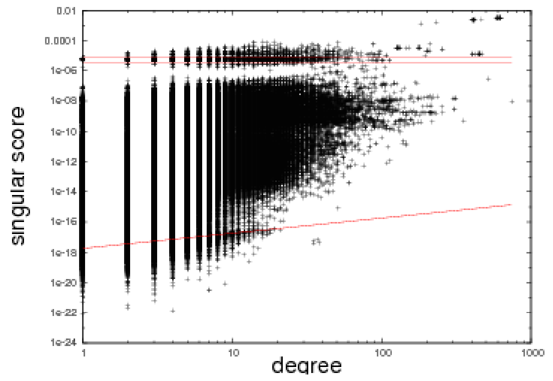


(SDSD-plot)

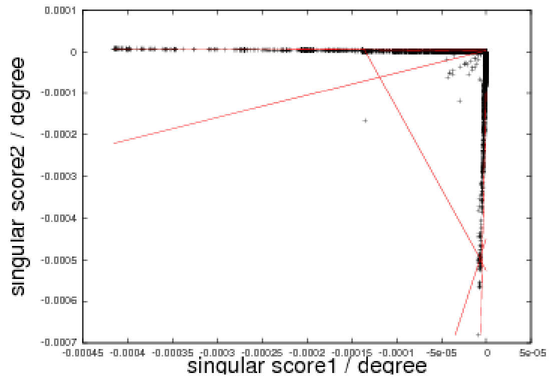
Figure 5.10: Results on IMDb (unweighted graph). Left DS-plot (first, and second singular score), and left SDSD-plot. Red lines indicate patterns detected by *EigenDiagnostics*. Details are in section 5.5.



(DS-plot of 1st singular score)

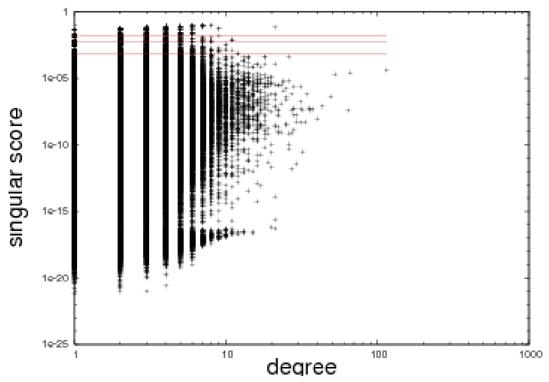


(DS-plot of 2nd singular score)

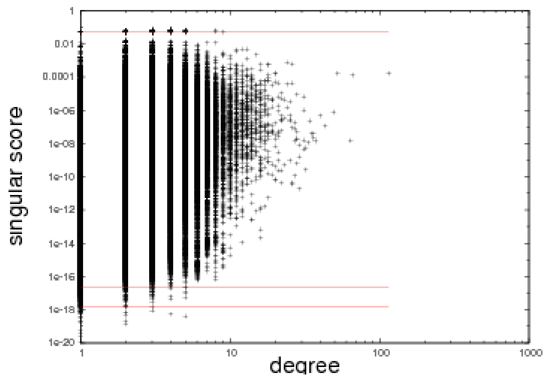


(SDSD-plot)

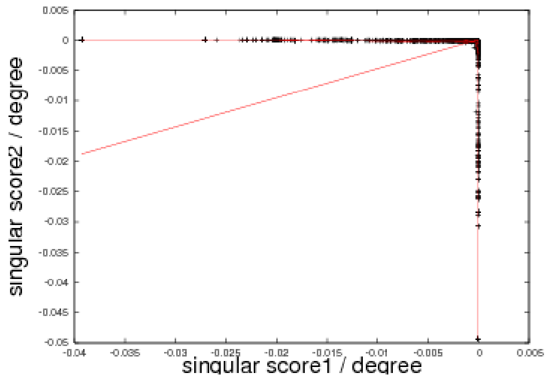
Figure 5.11: Results on Patent Citation(unweighted graph). Left DS-plot (first, and second singular score), and left SDSD-plot. Red lines indicate patterns detected by *EigenDiagnostics*. Details are in section 5.5.



(DS-plot of 1st singular score)

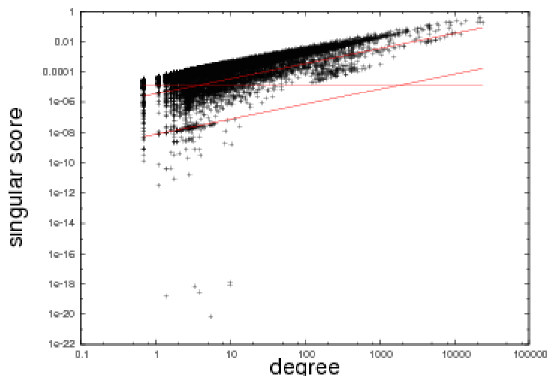


(DS-plot of 2nd singular score)

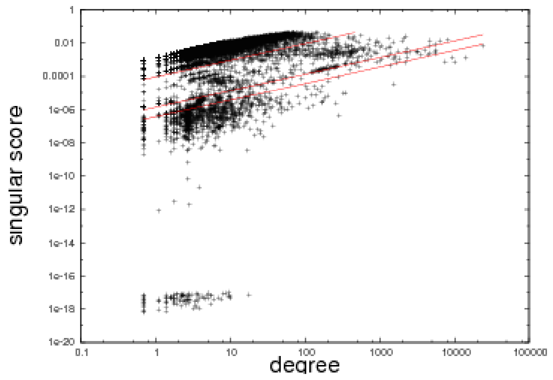


(SDSD-plot)

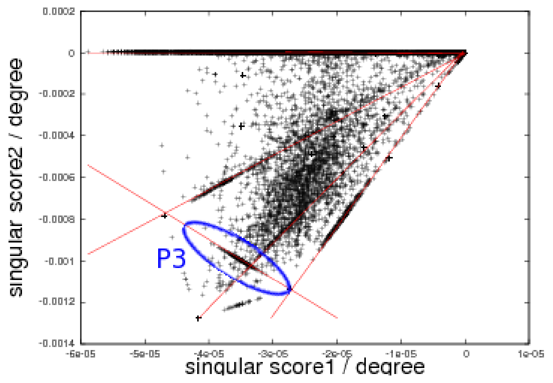
Figure 5.12: Results on DBLP (unweighted graph). Left DS-plot (first, and second singular score), and left SDSD-plot. Red lines indicate patterns detected by *EigenDiagnostics*. Details are in section 5.5.



(DS-plot of 1st singular score)

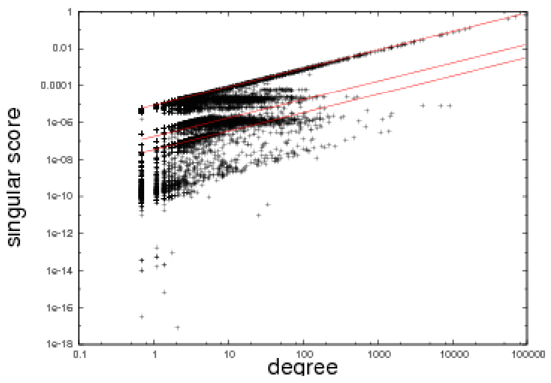


(DS-plot of 2nd singular score)

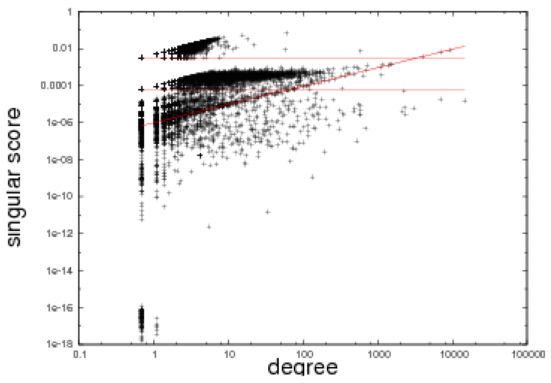


(SDSD-plot)

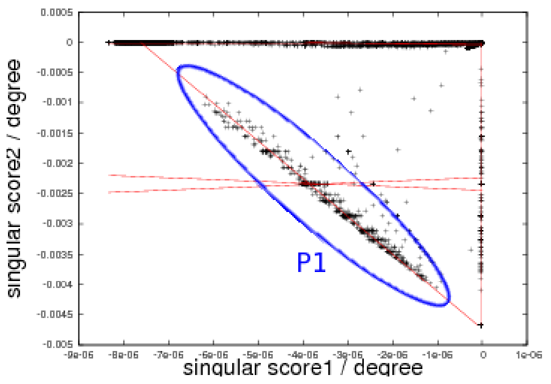
Figure 5.13: Results on LBNL regular TCP IP-source vs port-number (LBNL-sp) (weighted graph). Left DS-plot (first, and second singular score), and left SDSD-plot. Red lines indicate patterns detected by *EigenDiagnostics*. Details are in section 5.5.



(DS-plot of 1st singular score)

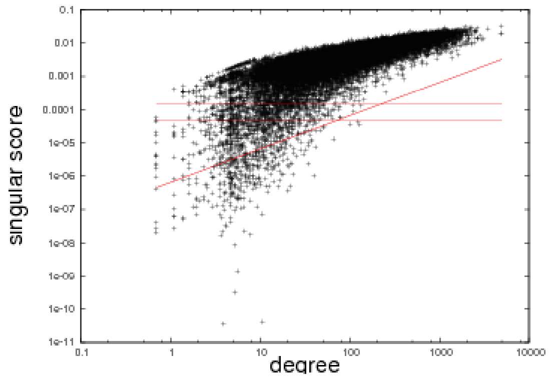


(DS-plot of 2nd singular score)

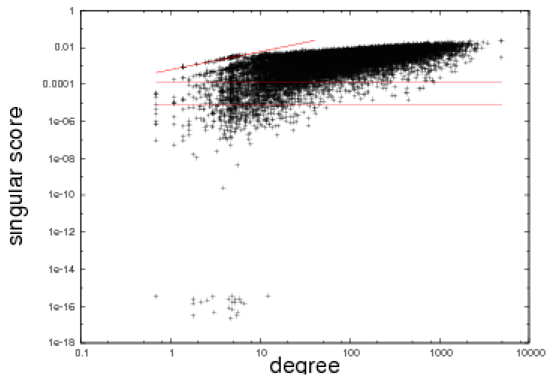


(SDSD-plot)

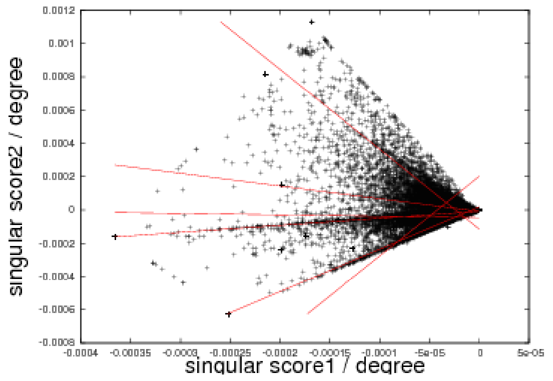
Figure 5.14: Results on LBNL scanners UDP IP-destination vs port-number (LBNL-dp) (weighted graph). Left DS-plot (first, and second singular score), and left SDSD-plot. Red lines indicate patterns detected by *EigenDiagnostics*. Details are in section 5.5.



(DS-plot of 1st singular score)



(DS-plot of 2nd singular score)



(SDSD-plot)

Figure 5.15: Results on Click Stream(weighted graph). Left DS-plot (first, and second singular score), and left SDSD-plot. Red lines indicate patterns detected by *EigenDiagnostics*. Details are in section 5.5.

Chapter 6

Shortest Path Distance Estimation Using Eigen Decomposition

Estimating the distances of the shortest path between given pairs of nodes in a graph is a basic operation in a wide variety of applications including social network analysis, web retrieval, etc. Such applications require a response on the order of a few milliseconds, but *exact* algorithms to compute the distance of the shortest path exactly do not work on real-world large-scale networks, because of their infeasible time complexities. The landmark-based methods approximate distances by using a few nodes as landmarks, and can accurately estimate shortest-path distances with feasible time complexities. However, they fail at estimating small distances, as it is difficult for a few selected landmarks to cover the shortest paths of many close node pairs.

To tackle this problem, we present a novel method EigenSP, that estimates the shortest-path distance by using an adjacency matrix approximated by a few eigenvalues and eigenvectors. As discussed in Section 3.2, we can expect to estimate distances of node pairs located nearby by using eigen decomposition. The average relative error rate of EigenSP is lower than that of the landmark-based methods on large graphs with many short distances. Empirical results suggest that EigenSP estimates small distances better than the landmark-based methods.

This work was published in [112].

6.1 Introduction

Estimating the distance of the shortest path between given pairs of nodes in a graph is a basic operation in a wide variety of applications including social network analysis [163], web retrieval [161], etc. Such applications require responses on the order of a few milliseconds, yet *exact* algorithms to compute the shortest-path distance exactly [48] fail on real-world large scale networks, due to their infeasible time complexities. Precomputing all the shortest paths and storing them is a straightforward solution, but this is infeasible since it would require storage of an extremely huge matrix.

The landmark-based methods can accurately estimate shortest-path distance with feasible space and time complexities [134, 91]. These methods approximate shortest-path distance by analyzing the relationship between a given node pair and a few selected nodes as *landmarks* [125, 152, 171, 172, 108, 136]. The landmark-based methods can accurately approximate network distances if the landmarks are properly selected [134, 106], but tend to err in approximating short distances. The reason for this failure is that it is difficult for a few landmarks to cover the many shortest paths of close node pairs in a large graph. This is an unwanted property for applications which try to identify *potential friends* located nearby, as is done in social network analysis. For example, in people search, a user want to sort the list of people retrieved by some keywords according to the distance from the user in social network. In this case, a direct friend of the user should be ranked in the top of the list, and people of distance 2, *i.e.*, friends of friends, should be ranked after them. But longer distances are not so important, that is, it does not matter whether people in the list are distance 5 or distance 7, because both of them would be remotely related to the user in social network. Thus, in some applications, estimating short distances accurately is critical, while accurate estimation is less important over long distances.

In this chapter, we present a novel method, EigenSP. Our intuition is that the shortest-path distance can be estimated as the smallest number of k such that values in the k-times multiplied adjacency matrix corresponding to node pairs are more than 0, and that the values can be approximated by using a few eigenvalues and eigenvectors of the original adjacency matrix. Approximation error is expected to be small for a smaller value of k, which means that the estimation error of the shortest-path distance is expected to be small for close node pairs. Our method can estimate short distances better than the landmark-based methods. To the best of our knowledge, this is the first analysis of the feasibility and efficiency of estimating shortest-path distances with an adjacency matrix approximated by a few eigenvalues and eigenvectors. Our main contributions are summarized as follows:

- We propose a novel method, EigenSP, that can accurately approximate the distance in a graph with good scalability on the basis of the spectral theorem. We also propose an improvement of EigenSP with a simple heuristic (EigenSP-M) to deal with a problem that arises with EigenSP.
- We show that our method is more accurate than the landmark-based methods, especially in estimating short distances, on eight different large real-world networks.

The remainder of this chapter is organized as follows: we describe the related literatures in Section 6.2, and introduce our method in Section 6.3. We describe the experimental evaluation in Section 6.4, while Section 6.5 concludes the discussion.

6.2 Related Work

The landmark-based methods have been widely used for shortest path distance estimation as we discussed in Section 2.3. They select the landmark nodes according to several centrality measures (see Section 2.3). However, landmark-based methods tend to err in approximating short distances, since it is difficult for a few landmarks to cover the many shortest paths of close node pairs in a large graph. We tackle this problem by using an adjacency matrix approximated by a small number of eigenvalues and eigenvectors.

6.3 Proposed Method

6.3.1 Problem Definition

Given an unweighted, undirected graph $G = (V, E)$ with no self-edges, the shortest path distance between node i and node j $(i, j \in V)$ is the number of edges within the shortest path. The distance is considered to be infinite when there is no path between node i and node j (disconnected).

Our problem can be summarized as follows:

Problem 3 given a node pair, how can we estimate shortest-path distance within the order of microseconds for query time, with feasible space requirements, and good time complexity for pre-computing?

Table 6.1 shows the notation used in this chapter in addition to Table 3.1 and Table 3.2.

Symbol	Definition
D	diameter (longest shortest path distance) of G
$dist(i,j)$	shortest path distance between node i and node j
$dist'(i,j)$	estimated $dist(i,j)$
$a_{ij}^{(k)}$	i-th row and j-th column element of A^k
A_R	approximation of A $(= X_R \Lambda_R X_R^T)$
$a'^{(R,k)}_{ij}$	i-th row and j-th column element of A_R^k
$t^{(R,k)}$	threshold to guess $a_{ij}^{(k)}$ using $a'^{(R,k)}_{ij}$
$t_a^{(R,k)}$	threshold to guess $a_{ij}^{(k)}$ using $R^{-1} \sum_{r=1}^{R} a'^{(r,k)}_{ij}$

Table 6.1: Symbol table (in addition to Table 3.1 and Table 3.2)

6.3.2 Computing Distances by Eigenvectors

Here we show a way to compute the shortest-path distance by using eigenvectors. The shortest-path distance and the eigenvalue and eigenvectors of the adjacency matrix are related by the following theorem.

Theorem 1 Let G be an undirected, simple graph and A be the adjacency matrix representation of it. The shortest path distance $dist(i,j)$ between two distinct nodes i and node j satisfies the following equation:

$$dist(i,j) = \min \left\{ k \,\middle|\, \sum_{r=1}^{n} x_{ri} \lambda_r^k x_{rj} > 0, k \in \mathcal{N} \right\} \tag{6.1}$$

where x_{ri} is the i-th element of the r-th eigenvector, λ_r is the r-th eigenvalue of the adjacency matrix, and n is the number of orthogonal eigenvectors.

Proof 1 Since G is undirected, A is a real, symmetric matrix. Thus, from the spectral theorem, we can diagonalize A by using its eigenvalues and eigenvectors. Therefore, $A = X \Lambda X^T$, where Λ is a diagonal matrix containing the eigenvalues of A and $X = [x_1 | \dots | x_n]$ is an orthonormal matrix containing in its i-th column the eigenvector x_i corresponding to the i-th eigenvalue $\lambda_i, i = 1, \dots, n$. From the orthonormality of X, it follows that $A^k = X \Lambda^k X^T$ $(\diamondsuit)$.

Now we have the following lemma:

Lemma 1 (Algebraic path counting) Let $a_{ij}^{(k)}$ be the i-th row and j-th column element of A^k. Then $a_{ij}^{(k)}$ equals the number of paths from node i to j of length exactly k.

If there are no paths from node i to j of length k, $a_{ij}^{(k)}$ is 0. From equation ($\diamondsuit$), it follows that $a_{ij}^{(k)} = \sum_{r=1}^{n} x_{ri}\lambda_r^k x_{rj}$. Combining these two facts, we obtain equation 6.1.

Now we can compute the shortest-path distance exactly by finding a k which satisfies equation 6.1, though we cannot directly adopt this as computing all eigenvalues and eigenvectors is infeasible for massive graph datasets.

6.3.3 Estimating Distances by Eigenvectors

The idea behind our method is to determine k that satisfies equation 6.1 by using ${a'}_{ij}^{(R,k)} = \sum_{r=1}^{R} x_{ri}\lambda_r^k x_{rj}$ with a small R. $A_R = X_R \Lambda_R X_R^T$ is an approximated matrix of A, and ${a'}_{ij}^{(R,k)}$ is the i-th row and j-th column element of $A_R^k = X_R \Lambda_R^k X_R^T$.

Based on the above idea we develop a novel method, EigenSP. EigenSP requires precomputed values of X_R and Λ_R of A with the desired rank R, and the thresholds $t^{(R,k)}$ determined by the distribution of sampled node pairs. When EigenSP receives a request to estimate the shortest-path distance of a node pair, it searches for the minimum k for which ${a'}_{ij}^{(R,k)}$ exceeds a threshold $t^{(R,k)}$. If ${a'}_{ij}^{(R,k)}$ does not exceed $t^{(R,k)}$ for any $k \leq D$, i.e., the diameter of the graph, it concludes that the nodes of the pair are disconnected. Algorithm 5 describes this procedure.

The time complexity of the distance estimation is on the order of the number of eigenvectors multiplied by the diameter, i.e., $O(RD)$. The space requirement of the distance estimation is dominated by holding the R eigenvectors, i.e., $O(R|V|)$. The time complexity and space requirements of precomputation are dominated by the eigenvalue computation of the sparse matrix, leveraged by several eigenvalue computation techniques such as the Lanczos algorithm [40].

6.3.4 How to Determine Thresholds

Finding a k satisfying equation 6.1 depends on how well we can distinguish node pairs with $a_{ij}^{(k)} = 0$ and those with $a_{ij}^{(k)} > 0$ for every k using ${a'}_{ij}^{(R,k)}$. There may be two types of errors:

Algorithm 5 EigenSP

```
Input: Node pair $i$ and $j$ to compute distance
Input: $X_R, \Lambda_R$
Input: Thresholds $t^{(R,k)}$
Input: Diameter $D$ of the graph
Output: Estimated distance
1: for $k = 1, \dots, D$ do
2:     if $\sum_{r=1}^{R} x_{ri}\lambda_r^k x_{rj} \geq t^{(R,k)}$ then
3:         return $k$
4:     end if
5: end for
6: return $\infty$
```

- (Type 0 error) $a_{ij}^{(k)}$ is 0 but estimated to be more than 0 ($a'^{(R,k)}_{ij} \geq t^{(R,k)}$).
- (Type 1 error) $a_{ij}^{(k)}$ is more than 0 but estimated to be 0 ($a'^{(R,k)}_{ij} < t^{(R,k)}$).

There is a trade-off between these types of error. EigenSP tends to estimate the distances to be shorter than the actual distances if the type 0 error rate dominates the type 1 error rate, and vice versa. The simple way to determine $t^{(R,k)}$ is to sample node pairs and determine $t^{(R,k)}$ that minimizes the error rate of both types for each k. But we cannot adopt this simple sampling method, as the number of connected node pairs with $a_{ij}^{(k)} > 0$ is very small for small values of k, and those with $a_{ij}^{(k)} = 0$ rapidly decrease in number when k grows. We show the number of connected node pairs in Enron dataset described in Section 6.4 (Figure 6.1). So we cannot have enough sample node pairs to determine $t^{(R,k)}$ for a small and large k.

Then how do we determine thresholds for those small and large k? Figure 6.2 shows thresholds with several error rates of each error type for k with enough sampled node pairs. Thresholds with similar type 0 error rates or thresholds with similar type 1 error rates exponentially increase as k grows. The most likely explanation for this is that the error $\sum_{r=R+1}^{n} x_{ri}\lambda_r^k x_{rj}$ exponentially grows at a rate proportional to the dominated λ_r ($r = R+1, \dots, n$), resulting in a type 0 error. In addition, the *exact* value $\sum_{r=1}^{n} x_{ri}\lambda_r^k x_{rj}$ also exponentially grows at a larger rate proportional to the dominated λ_r ($r = 1, \dots, n$), so thresholds with similar type 1 error rates will diverge away from zero as k grows. According to this observation, we model thresholds

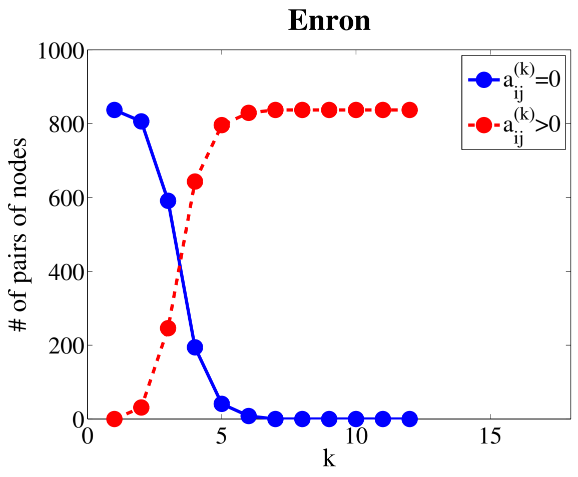
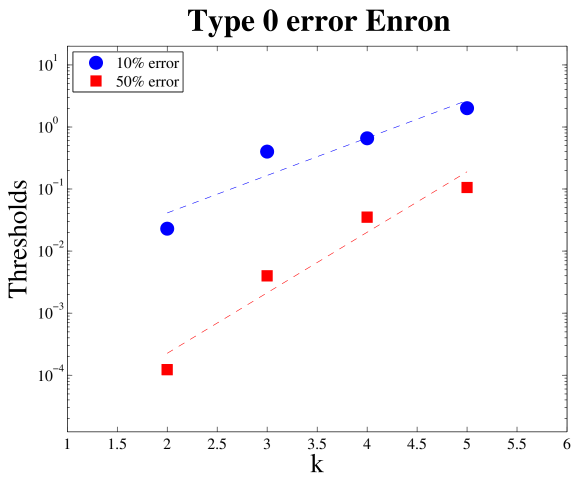


Figure 6.1: The number of connected node pairs with $a_{ij}^{(k)} = 0$ or $a_{ij}^{(k)} > 0$ among 1,000 sampled node pairs. Horizontal axis: k. Vertical axis: the number of connected node pairs with $a_{ij}^{(k)} = 0$ or $a_{ij}^{(k)} > 0$.

as $t^{(R,k)} = \beta\alpha^k$ so that type 0 error rates or type 1 error rates are almost constant for every k. The ideal ratio of type 0 error rate to type 1 error rate depends on the application. We adopt the ratio of one through which the errors of both shorter and longer distances than the actual distances are minimized to the same extent. To determine α and β, first we calculate thresholds for k for which there are enough node pairs with $a_{ij}^{(k)} = 0$ and $a_{ij}^{(k)} > 0$ within 1,000 sampled node pairs, such that error rates of type 0 and type 1 are nearly equal. Next we calculate the slope a and intercept b for the least squares regression line for the relationship between the logarithm of the thresholds and k, and determine α as 10^a and β as 10^b. For Enron data, α is 3.47 and β is $0.60 * 10^{-2}$.

To clarify for which node pairs it is easy to guess that $a_{ij}^{(k)}$ is 0 or not, we also examine the relation of $a_{ij}^{(k)}$ and ${a'}_{ij}^{(R,k)}$. Figure 6.3 compares $a_{ij}^{(3)}$ and ${a'}_{ij}^{(10,3)}$, with the plot showing a strong correlation between them. The same is true for any values of k and R. This means that the bigger $a_{ij}^{(k)}$ is, the easier it is to guess whether $a_{ij}^{(k)}$ is 0 or more than 0.



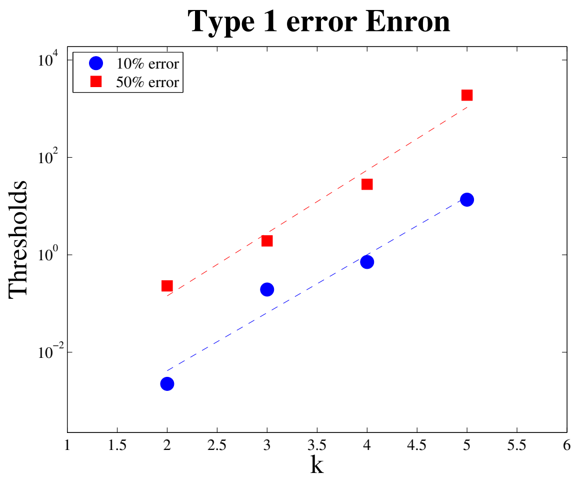


Figure 6.2: Thresholds with several error rates of each error types for k with enough sampled node pairs. Horizontal axis: k. Vertical axis: thresholds with error rates of 10% and 50%.

6.3.5 Estimating Distances More Accurately

We find that the ${a'}_{ij}^{(R,k)}$ become unstable as the number of eigenvectors R grows. In other words, ${a'}_{ij}^{(R,k)}$ does not always approach $a_{ij}^{(k)}$ rapidly enough as R increases. Figure 6.4 plots ${a'}_{ij}^{(R,3)}$ and ${a'}_{ij}^{(R,4)}$ of an example node pair of Enron dataset for $R = 1, \ldots, 500$ (blue line). $dist(i, j)$ is 4 because $a_{ij}^{(3)} = 0$ and $a_{ij}^{(4)} = 24 > 0$. For some R , ${a'}_{ij}^{(R,3)}$ may exceed $t^{(R,3)}$ and $dist'(i, j)$ may

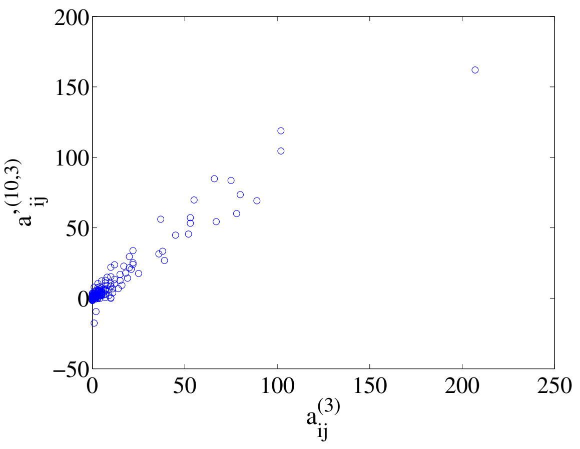


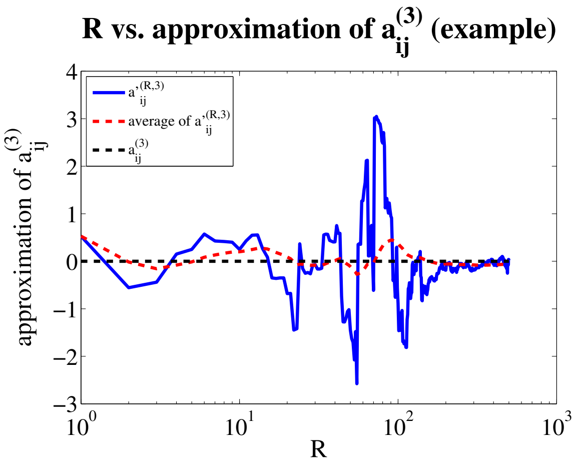
Figure 6.3: $a_{ij}^{(3)}$ (horizontal axis) are highly correlated with $a'^{(10,3)}_{ij}$ (vertical axis) on Enron graph. These two values are also highly correlated for other values of k and R(not shown).

be smaller than $dist(i,j)$. But $dist'(i,j)$ may be larger than $dist(i,j)$ for other R, because $a'^{(R,4)}_{ij}$ may still be negative. In general, even if $a'^{(R,k)}_{ij}$ is very close to $a_{ij}^{(k)}$ for some R, it may still be much further away for a larger R. This instability may cause errors in distance estimations.

One possible explanation for this instability of $a'^{(R,k)}_{ij}$ can be found in the distribution of eigenvalues (Figure 6.5). The absolute values of the eigenvalues decrease very slowly, and even the 500th largest eigenvalue is not much smaller than the 1st eigenvalue. This means the elements of the R-th eigenvector can strongly affect $a'^{(R,k)}_{ij}$, even if R is about 500.

To deal with this problem, we add an easy heuristic to EigenSP (we call this modification, EigenSP-M). To find whether $a_{ij}^{(k)}$ is 0 or not, EigenSP-M uses the average value of $a'^{(r,k)}_{ij}$ ($r = 1, \ldots, R$), *i.e.*, $R^{-1}\sum_{r=1}^{R} a'^{(r,k)}_{ij}$, instead of $a'^{(R,k)}_{ij}$ itself. Figure 6.4 shows the $R^{-1}\sum_{r=1}^{R} a'^{(r,k)}_{ij}$ for $R = 1, \ldots, 500$ (red dotted line). These values are more stable and are closer to $a_{ij}^{(k)}$ for many R. With this heuristic, EigenSP-M should be able to reduce the error.

To show efficiency of this heuristic more generally, we show mean error (ME) and mean squared error (MSE) of approximation for k as $a_{ij}^{(k-1)} = 0$ and $a_{ij}^{(k)} > 0$ among 1,000 sampled node pairs in Enron dataset (Figure 6.6). These errors should directly affect the accuracy of EigenSP. ME of $a'^{(R,k-1)}_{ij}$



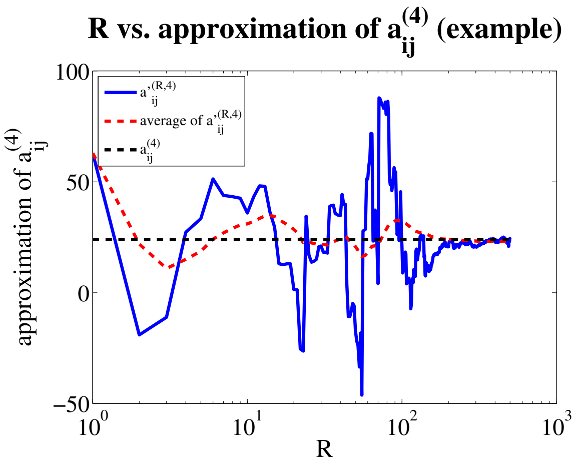


Figure 6.4: $a'^{(R,k)}_{ij}$ of an example element has a wide variety of values which do not depend on the number of R (blue line). In this sample on an Enron dataset, $a^{(k)}_{ij}$ is 0 ($k = 3$) and 24 ($k = 4$) (black dotted straight line). $R^{-1}\sum_{r=1}^{R} a'^{(r,k)}_{ij}$ does not vary much and is in fact closer to $a^{(k)}_{ij}$ (red dotted line). (top)$k = 3$. (bottom)$k = 4$. Horizontal axis: the number of R. Vertical axis: $a^{(k)}_{ij}$ or $a'^{(R,k)}_{ij}$ or $R^{-1}\sum_{r=1}^{R} a'^{(r,k)}_{ij}$.

and $a'^{(R,k)}_{ij}$ totally decrease as R grows, but have a wide variety of values which do not depend on the number of R (blue lines in Figure 6.6(a)(c)). Moreover, MSE of $a'^{(R,k-1)}_{ij}$ and $a'^{(R,k)}_{ij}$ are high not depending on the number of R (blue lines in Figure 6.6(b)(d)). These indicate the accuracy of

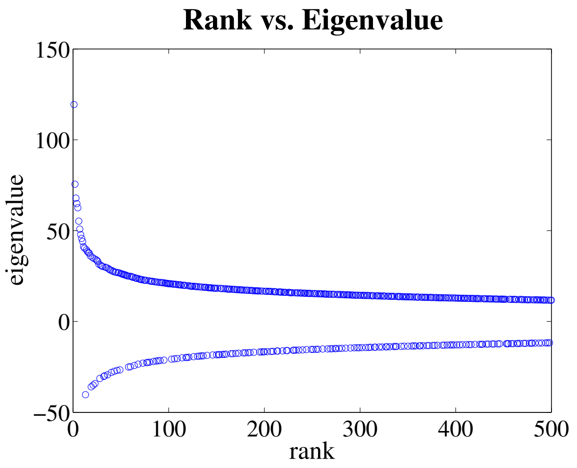
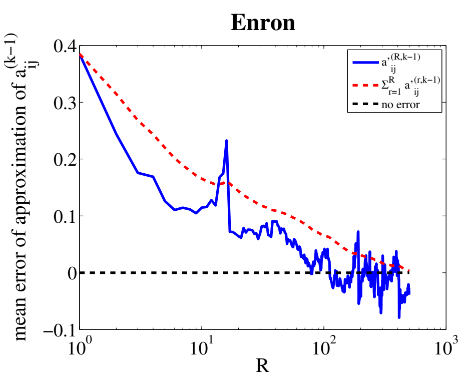


Figure 6.5: Distribution of eigenvalues of the adjacency matrix of Enron dataset. The absolute values of the 500-th eigenvalue (11.7) was not much smaller than that of the 1-st eigenvalue (119.4).

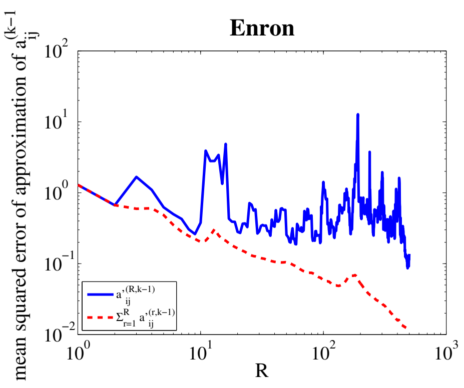
EigenSP differs so much according to the choise of R. On the other hand, ME of $R^{-1}\sum_{r=1}^{R} {a'}_{ij}^{(r,k-1)}$ and $R^{-1}\sum_{r=1}^{R} {a'}_{ij}^{(r,k)}$ are higher than ME of ${a'}_{ij}^{(R,k-1)}$ and ${a'}_{ij}^{(R,k)}$, but smoothly decrease as R grows (red dotted lines in Figure 6.6(a)(c)). Moreover, MSE of $R^{-1}\sum_{r=1}^{R} {a'}_{ij}^{(r,k-1)}$ and $R^{-1}\sum_{r=1}^{R} {a'}_{ij}^{(r,k)}$ decrease as R grows (red dotted lines in Figure 6.6(b)(d)). These indicate the accuracy of EigenSP-M should smoothly become better as R grows.

In addition, we show those in CondMat dataset also described in Section 6.4 (Figure 6.7). In CondMat dataset, ME of ${a'}_{ij}^{(R,k-1)}$ and ${a'}_{ij}^{(R,k)}$ do not decrease as R grows, and even MSE increase as R grows (blue lines in Figure 6.7). On the other hand, ME of $R^{-1}\sum_{r=1}^{R} {a'}_{ij}^{(r,k-1)}$ and $R^{-1}\sum_{r=1}^{R} {a'}_{ij}^{(r,k)}$ decrease as R grows, and MSE does not increase as R grows (red dotted lines in Figure 6.7). These indicate that the accuracy of EigenSP can become worse if we use larger R, and EigenSP-M should suppress such instability, and be better than EigenSP.

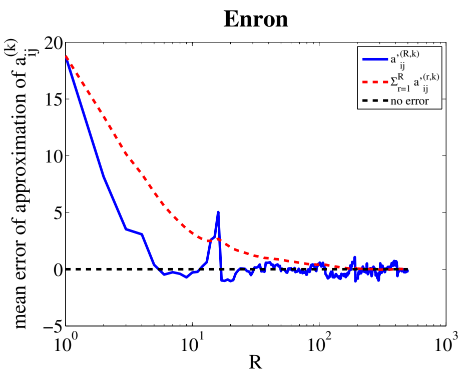
As an important observation, we also show mean error (ME) of approximation for $k = 2, \ldots, 5$ among 1,000 sampled node pairs in Enron dataset in Figure 6.8. Even though accuracies for individual node pairs are unstable for the number of R as shown in Figure 6.4, errors in average seem to decrease as R grows. Moreover, we show mean squared error (MSE) in Figure 6.9. Roughly speaking, MSE decrease linearly in these log-log plots,



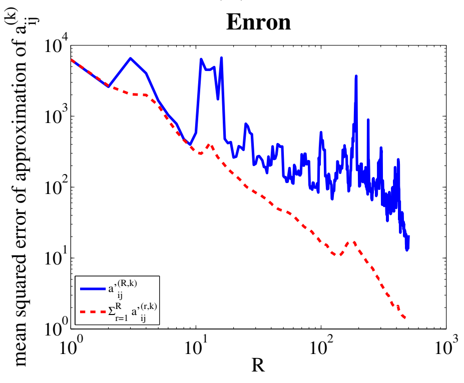
(a)



(b)



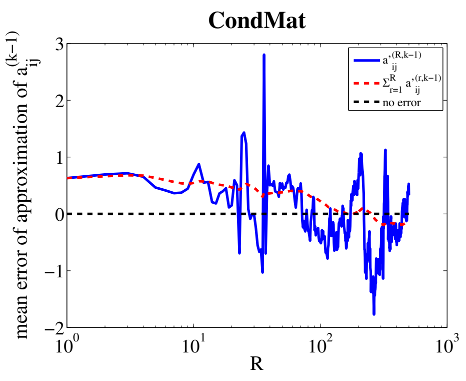
(c)



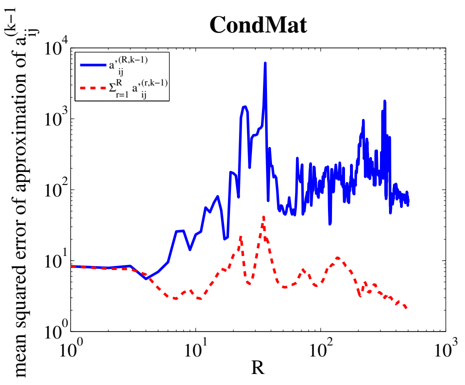
(d)

Figure 6.6: Mean error (ME) and mean squared error (MSE) of approximation for k as $a_{ij}^{(k-1)} = 0$ and $a_{ij}^{(k)} > 0$ among 1,000 sampled node pairs in Enron dataset. (a) ME of approximation of $a_{ij}^{(k-1)}$, (b) MSE of approximation of $a_{ij}^{(k-1)}$, (c) ME of approximation of $a_{ij}^{(k)}$, and (d) MSE of approximation of $a_{ij}^{(k)}$. Blue line: ME and MSE of $a'^{(R,k-1)}_{ij}$ and $a'^{(R,k)}_{ij}$. Red dotted line: ME and MSE of $R^{-1}\sum_{r=1}^{R} a'^{(r,k-1)}_{ij}$ and $R^{-1}\sum_{r=1}^{R} a'^{(r,k)}_{ij}$. Black dotted straight line: line of no error. Horizontal axis: the number of R. Vertical axis: ME in (a)(c), and MSE in (b)(d).

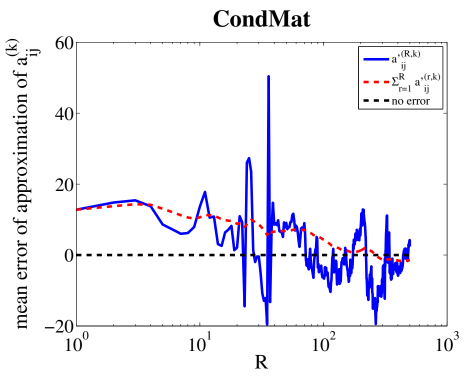
which indicates that error rate is proportional to some powers of R. This is analogous to a Monte Carlo method in which random sampling is repeated R times, and error rate decreases roughly linearly to $1/\sqrt{N}$ where $N(=R)$ is the number of sampled points. Thus, error rate in average decreases as R grow in view of specific value of k, whereas error rate for k as $a_{ij}^{(k-1)} = 0$



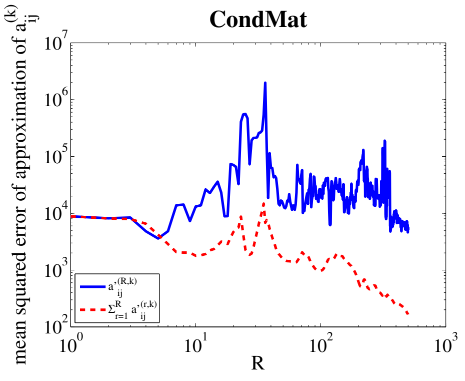
(a)



(b)



(c)

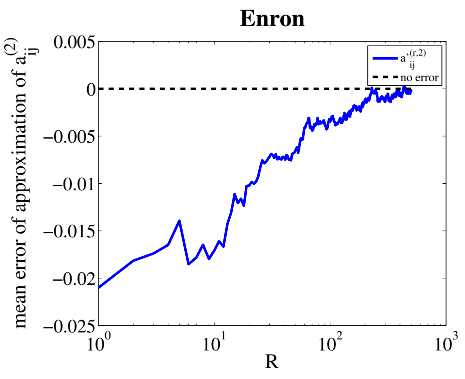


(d)

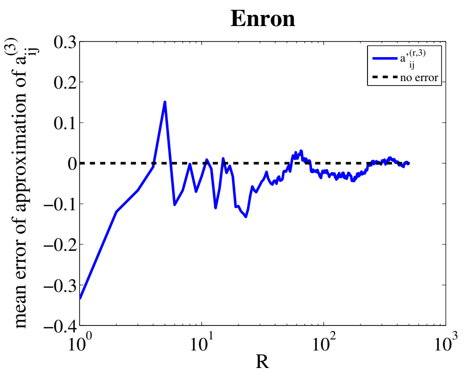
Figure 6.7: Mean error (ME) and mean squared error (MSE) of approximation for k as $a_{ij}^{(k-1)} = 0$ and $a_{ij}^{(k)} > 0$ among 1,000 sampled node pairs in CondMat dataset. (a) ME of approximation of $a_{ij}^{(k-1)}$, (b) MSE of approximation of $a_{ij}^{(k-1)}$, (c) ME of approximation of $a_{ij}^{(k)}$, and (d) MSE of approximation of $a_{ij}^{(k)}$. Blue line: ME and MSE of $a'^{(R,k-1)}_{ij}$ and $a'^{(R,k)}_{ij}$. Red dotted line: ME and MSE of $R^{-1}\sum_{r=1}^{R} a'^{(r,k-1)}_{ij}$ and $R^{-1}\sum_{r=1}^{R} a'^{(r,k)}_{ij}$. Black dotted straight line: line of no error. Horizontal axis: the number of R. Vertical axis: ME in (a)(c), and MSE in (b)(d).

and $a_{ij}^{(k)} > 0$ is unstable as shown in Figure 6.6. This makes the problem complicated, and this is why we need the heuristic we propose.

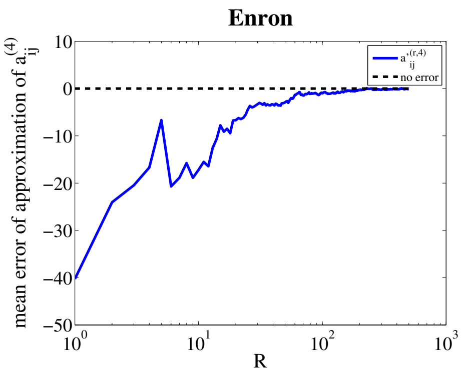
We also model thresholds for EigenSP-M as $t_a^{(R,k)} = \beta\alpha^k$, and determine α and β by using the same method as EigenSP to make the error rates of type 0 and type 1 equal. For Enron data, α is 2.58 and β is $1.72 * 10^{-2}$.



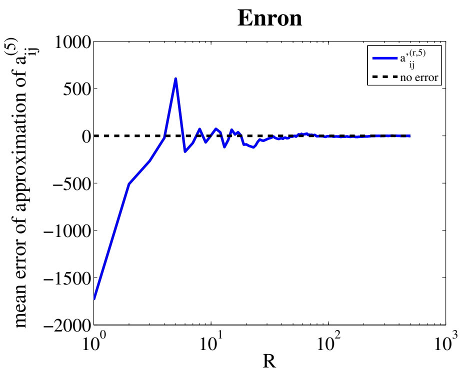
(a)



(b)



(c)



(d)

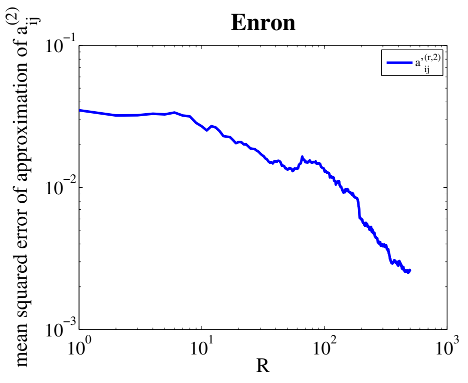
Figure 6.8: Mean error (ME) of approximation for $k = 2, \ldots, 5$ among 1,000 sampled node pairs in Enron dataset. (a) $k = 2$, (b) $k = 3$, (c) $k = 4$, and (d) $k = 5$. Blue line: ME of $a'^{(R,k)}_{ij}$. Black dotted straight line: line of no error. Horizontal axis: the number of R. Vertical axis: ME.

6.4 Empirical Results

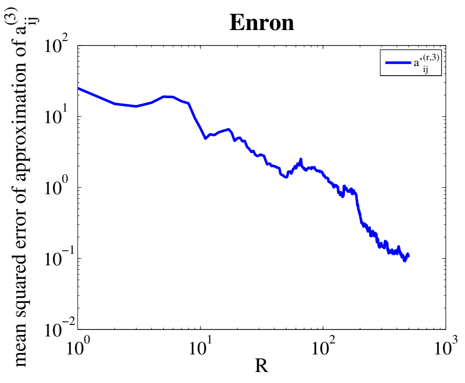
6.4.1 Data and Environment

We conduct experiments using eight real-world datasets in order to demonstrate the efficiency and scalability of our method. Six datasets (AS-CAIDA, Enron, AstroPh, CondMat, GrQc, HepTh) were taken from the Stanford Network Analysis Project (SNAP[1]). The other two datasets (DBLP, IMDb) were obtained from their respective websites. We use the diameters de-

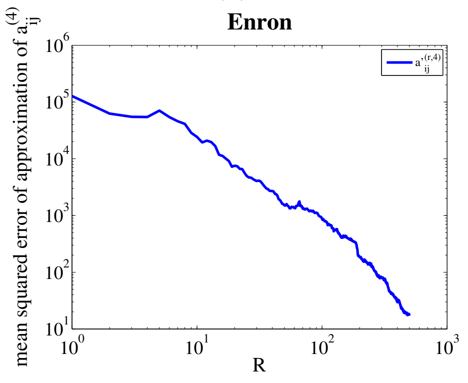
[1]http://snap.stanford.edu/



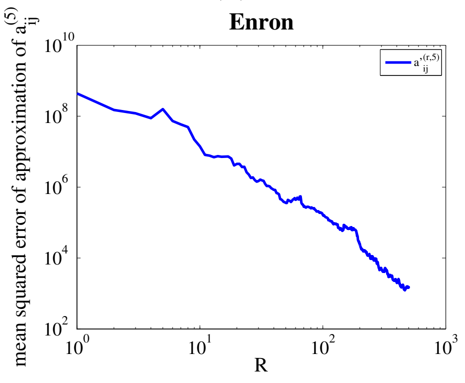
(a)



(b)



(c)



(d)

Figure 6.9: Mean squared error (MSE) of approximation for $k = 2, \ldots, 5$ among 1,000 sampled node pairs in Enron dataset. (a) $k = 2$, (b) $k = 3$, (c) $k = 4$, and (d) $k = 5$. Blue line: MSE of $a'^{(R,k)}_{ij}$. Horizontal axis: the number of R. Vertical axis: MSE.

scribed in the SNAP web site. We estimate the diameters of DBLP and IMDb by sampling over $1,000$ random nodes. Table 6.2 summarizes the data.

- **IMDb:** A collaboration network between actors, obtained from IMDb[2].
- **AS-CAIDA:** CAIDA AS graph from November 5 2007.
- **Enron:** Email communication network in a company.

[2]http://www.imdb.com

Dataset	$\|V\|$	$\|E\|$	D	$\overline{dist}$	P_{inf}
IMDb	551,200	54,544,140	13	3.67	0.02
AS-CAIDA	26,475	106,762	17	3.88	0.00
Enron	36,692	367,662	12	3.96	0.16
AstroPh	18,772	396,160	14	4.20	0.08
CondMat	23,133	186,936	15	5.35	0.15
GrQc	5,242	28,980	17	5.98	0.36
HepTh	9,877	51,971	17	5.98	0.23
DBLP	286,944	1,802,480	22	7.17	0.40

Table 6.2: A summary of datasets. $|V|$: # of nodes. $|E|$: # of edges. D: diameter. $\overline{dist}$: the average of the shortest-path distance. P_{inf}: the ratio of the disconnected node pairs. $\overline{dist}$ and P_{inf} are estimated by using $1,000$ randomly sampled node pairs.

- **AstroPh, CondMat, GrQc, HepTh:** Scientific collaborations in astro physics, condense matter physics, general relativity and quantum cosmology, high energy physics - theory.
- **DBLP:** A collaboration network between authors in computer science, obtained from DBLP[3].

EigenSP is implemented in the MATLAB language and all the experiments are performed on a 64-bit Windows XP machine with four 2.8GHz cores and 8GB of memory.

6.4.2 Compared Methods

We compute our methods by using the top 10 eigenvalues and eigenvectors , which can be stored within about 8GB memory for today's common graph data with 100 million nodes, assuming it needs 8 bytes(double) to store each score.

We compare our methods with the landmark-based methods (in this section, we call them the *landmark methods*). One of said methods is the triangulated heuristic [134, 125]. This heuristic selects the landmarks in a graph and calculates distances between the landmarks and all the nodes in the data. Then, the upper bounds of the distances are taken from the minimum length of the paths connecting node pairs via the landmarks, and

[3]http://dblp.uni-trier.de/

the lower bounds from the maximum length of paths assuming one node is on shortest paths connecting another node and the landmarks. Various weighted averages of the upper and lower bounds can be used as distance functions. We use the upper bounds as estimated distances since most of the previous studies showed that the upper bounds performed far better than the lower bounds and other weighted averages [134, 125]. We compare our methods with the triangulated heuristic of 100 landmarks, as required memory is almost the same as storing 10 eigenvectors, that is, 8 bytes for scores of eigenvectors (double) and 1 bytes for the distance between the landmarks and all the nodes (unsigned char). The distances between the landmarks and all other nodes are precomputed by using simple BFS.

Another landmark method we compare is Global Network Positioning (GNP) [125]. GNP models a network as a geometric space and characterizes the position of any node, which are optimized such that the overall error between the *exact* distances and the geometric distances is minimized by using the Simplex Downhill method [119]. First, GNP computes positions of the landmarks, and then computes positions of all other nodes so as to minimize the error of estimated distances from the landmarks. The cost of computing positions is very expensive, and Orion bootstrap [171] tackles this problem by dividing the landmarks into two groups. We calculate a simple GNP of 10-dimensional positions from the 16 landmarks, and calculate Orion by using 84 *additional* landmarks, resulting in 100 landmarks in total, the same setup as [171]. The reason we choose a dimensionality of 10 is that required memory is the same as storing 10 eigenvectors. Moreover, we regard errors of the geometric distances (g) of disconnected node pairs as $D+1-g$ if $g < D+1$, and as zero if $g \geq D+1$.

Among the several landmark selection strategies for the triangulated heuristic and GNP, we choose a strategy which selects the top k highest degree nodes such that each selected node is more than 1-hop away from every other selected node. We choose this landmark selection strategy due to its high accuracy on many datasets [134].

6.4.3 Evaluation Metric

We evaluate the accuracy with the average relative error (ARE). ARE is the average of the relative approximation error: $|dist'(i,j) - dist(i,j)|/dist(i,j)$. This metric is widely used in the study of shortest path distance estimations [134, 171]. However, it cannot be directly used if the dataset contains disconnected node pairs separated by a distance of ∞. Thus, we make the distance of disconnected node pairs $D+1$, 1-hop longer than the diameter of the graph. The reason for this is that the main interest of many applications is over short distances, where it is unimportant whether two nodes are disconnected or connected over a very long distance in the graph. This fits the aim of the metric of ARE. We measure the accuracy of each method in calculating the shortest path distances of $1,000$ randomly sampled pairs of nodes, which are different from node pairs used to determine thresholds for EigenSP and EigenSP-M.

6.4.4 Estimation Accuracy

Figure 6.10 shows the experimental results of each dataset. On IMDb, AS-CAIDA, Enron and AstroPh, the accuracies of our methods are better than those of the landmark methods, whereas our methods perform worse than the landmark methods on other datasets. We should realize that the former group of datasets have short average distances of about 4, whereas the latter have long average distances of about 6 or 7 (Table 6.2). We omit the results of Orion because AREs of Orion are worse than all the methods in every dataset.

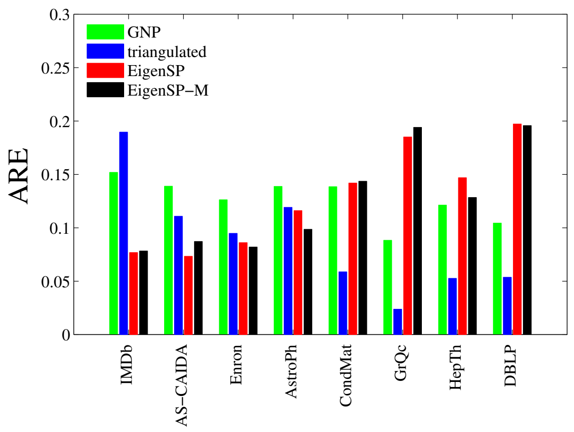


Figure 6.10: Average relative error (ARE) of each dataset. EigenSP and EigenSP-M can estimate distances better than the landmark method on datasets with short average distances of about 4 such as IMDb, AS-CAIDA, Enron, and AstroPh (see Table 6.2). Vertical axis: ARE.

What cause the difference in estimation accuracy among these methods? To answer this question, let us refer to the sum of relative errors (RE) of each dataset per the *exact* distance, that is, the sum of $|dist'(i,j) - dist(i,j)|/dist(i,j)$ for each $dist(i,j)$. On IMDb, AS-CAIDA, Enron and

AstroPh, on which the AREs of our methods are better than those of the landmark methods, our methods estimate many of the short distances more accurately than the landmark methods (Figure 6.11). Further, the triangulated heuristic has a high rate of error in estimating distances of 2 or 3, whereas GNP has large errors in estimating distances of 3 to 5, and the sum of RE of our methods are not severely different from or much smaller than those of the landmark methods in estimating these distances. Short average distances about 4 on these datasets (Table 6.2) can be caused by a large number of *hub* nodes in the graphs, and the landmark methods may have large errors due to a few landmarks being unable to cover these *hub* nodes.

On the other hand, the differences of the AREs on CondMat, GrQc, HepTh and DBLP between our methods and the landmark methods are mainly brought about when estimating the longer distances (Figure 6.12). Because these datasets contain many node pairs with long distances, that is, the average distance is about 6 or 7 (Table 6.2), a large amount of errors in our methods in these distances may have caused large AREs on these datasets.

The accuracies of EigenSP-M are almost as same as those of EigenSP with 10 eigenvalues. But EigenSP-M perform better than EigenSP with larger number of eigenvalues, especially on datasets of longer average distances, as we show in the following subsection.

EigenSP, EigenSP-M and GNP have high rates of error in disconnected node pairs. This type of error within EigenSP and EigenSP-M may be caused by the computational accuracy. For a disconnected node pair of i and j, either the score x_{ri} or x_{rj} in the r-th eigenvector must be theoretically zero, and ${a'}_{ij}^{(r,k)}$ should be zero for every r. But a small amount of absolute error in computing these x_{ri} or x_{rj} can occur and ${a'}_{ij}^{(r,k)}$ might exceed thresholds of the distances smaller than the diameter for large k. This kind of error can be reduced by calculating eigenvalues and eigenvectors more precisely. For GNP, this error may have been caused by our setup, in computing the positions of the disconnected node pairs, but this is outside the scope of this thesis.

6.4.5 Effect of Number of Eigenvalues

Next we show the accuracies of our methods with more than 10 eigenvalues. We show only the triangulated heuristic as the landmark method, because GNP took too long precomputing time to test with more than 10 landmarks.

On datasets which have short average distances, such as IMDb, AS-

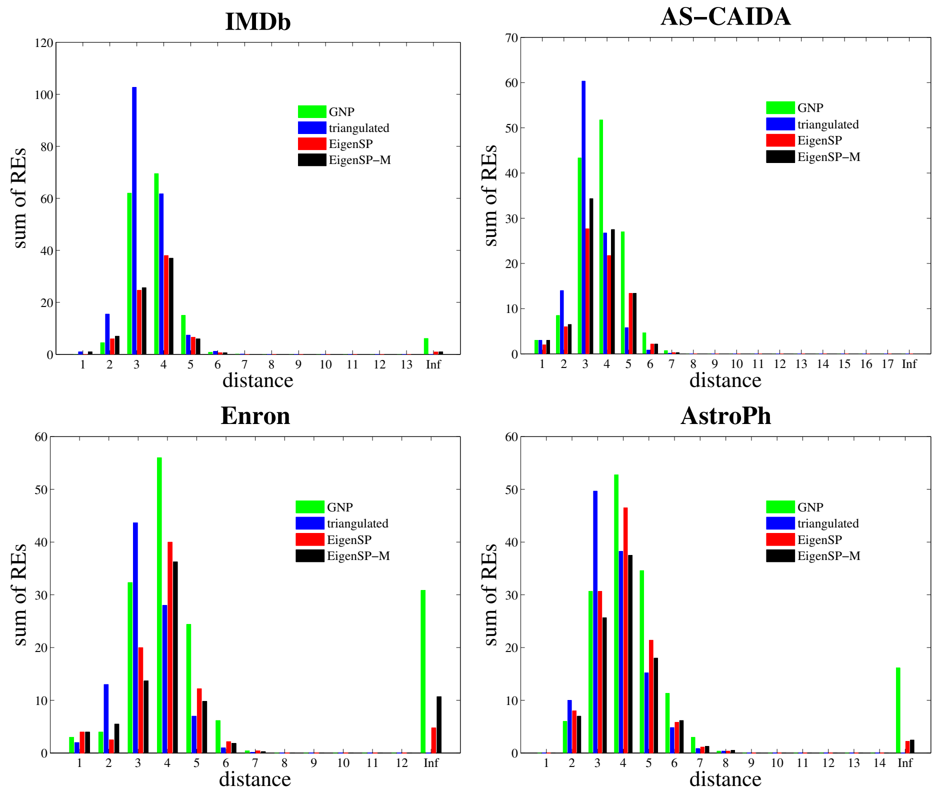


Figure 6.11: The sum of relative errors (RE) per the *exact* distance of datasets which have short average distances about 4 (Table 6.2). The triangulated heuristic has a high rate of error in estimating distances of 2 or 3, whereas GNP has large errors in estimating distances of 3 to 5, and the sum of RE of EigenSP and EigenSP-M are not severely different from or much smaller than those of the landmark methods in estimating these distances. Horizontal axis: *exact* distance. Vertical axis: the sum of REs.

CAIDA, Enron and AstroPh, both of EigenSP and EigenSP-M perform far better than the landmark method, even if the number of eigenvalues or landmarks grows(Figure 6.13).

On datasets which have long average distances, such as CondMat, GrQc, HepTh and DBLP, our method can not outperform the landmark method even with many eigenvalues(Figure 6.14). Interestingly, the accuracies of EigenSP-M are better than those of EigenSP if we use many eigenvalues on these datasets. Moreover, AREs of EigenSP with many eigenvalues can

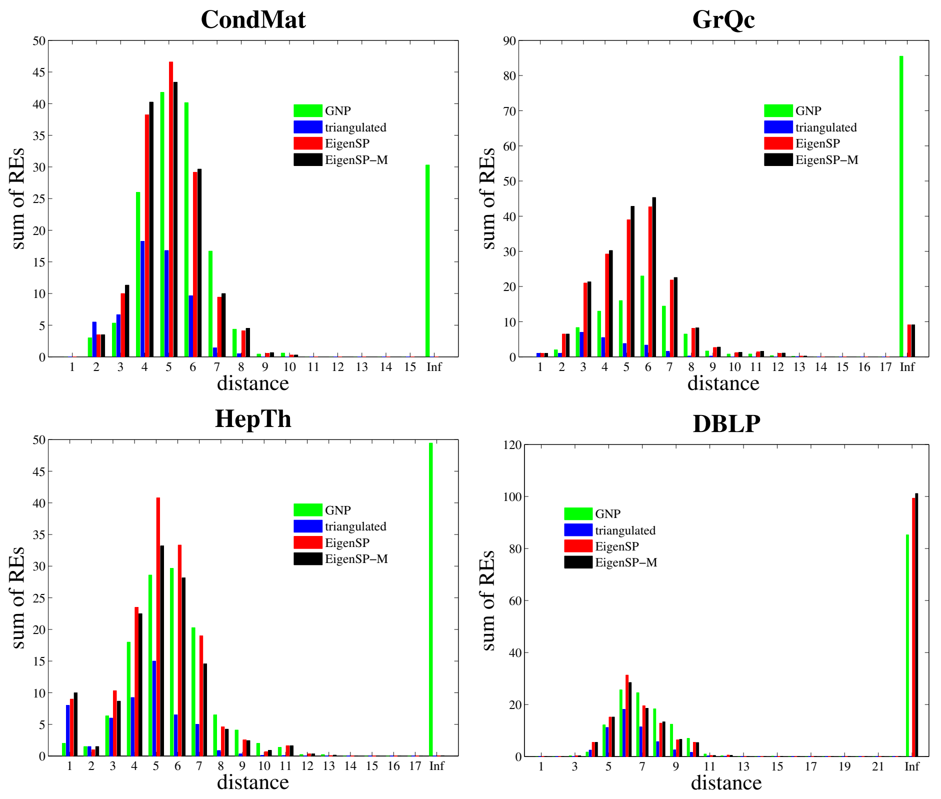


Figure 6.12: The sum of relative errors (RE) per the *exact* distance of datasets which had long average distances about 6 or 7 (Table 6.2). The differences of the AREs between our methods and the landmark methods are mainly brought about when estimating the longer distances. Horizontal axis: *exact* distance. Vertical axis: the sum of REs.

be worse than those with smaller number of eigenvalues, whereas AREs of EigenSP-M decrease as the number of eigenvalues grew. These results suggests that the heuristic of EigenSP-M can effectively overcome instability in estimating the values of the adjacency matrix for the variation of the number of eigenvalues.

6.4.6 Query Time

EigenSP and EigenSP-M can estimate for given node pairs in $O(RD)$, where R is the number of eigenvectors and D is the diameter of the graph. The triangulated heuristic can compute this in $O(L)$ where L is the number of

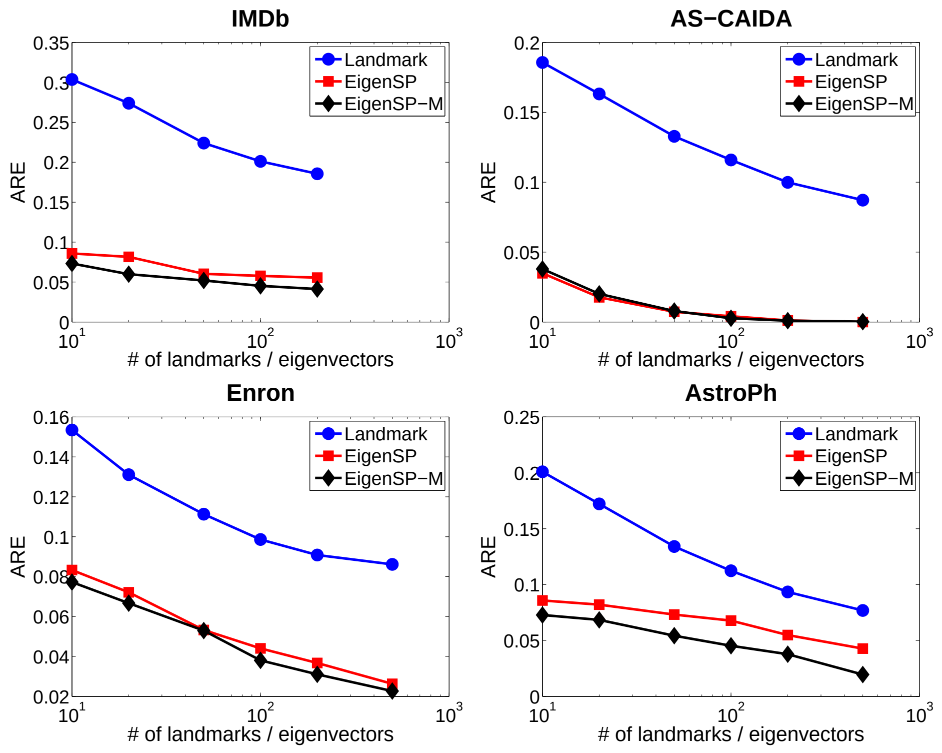


Figure 6.13: Average relative error (ARE) of datasets which have short average distances about 4 (Table 6.2) with many eigenvalues of landmarks. EigenSP and EigenSP-M can estimate distances far better than the landmark method can in these datasets, even with many eigenvalues or landmarks such as 200. Horizontal axis: the number of eigenvectors or landmarks. Vertical axis: ARE.

landmarks, and GNP in $O(S)$ where S is the dimensionality of the geometric space. All of the above time complexities are much improved over that of BFS, i.e., $O(|V| + |E|)$. Table 6.3 shows the average query time of these methods. All of these methods can estimate within the order of microseconds, much faster than BFS, which estimates within the order of milliseconds.

We also show the query time of BFS in estimating short distances in Figure 6.15, to answer a question such as "why not use BFS to estimate the short distances that is the advantage of EigenSP?" The query time of BFS increases almost linearly with the number of edges, and it takes more than $1ms$ for graphs with more than $100K$ edges, even in calculating distances

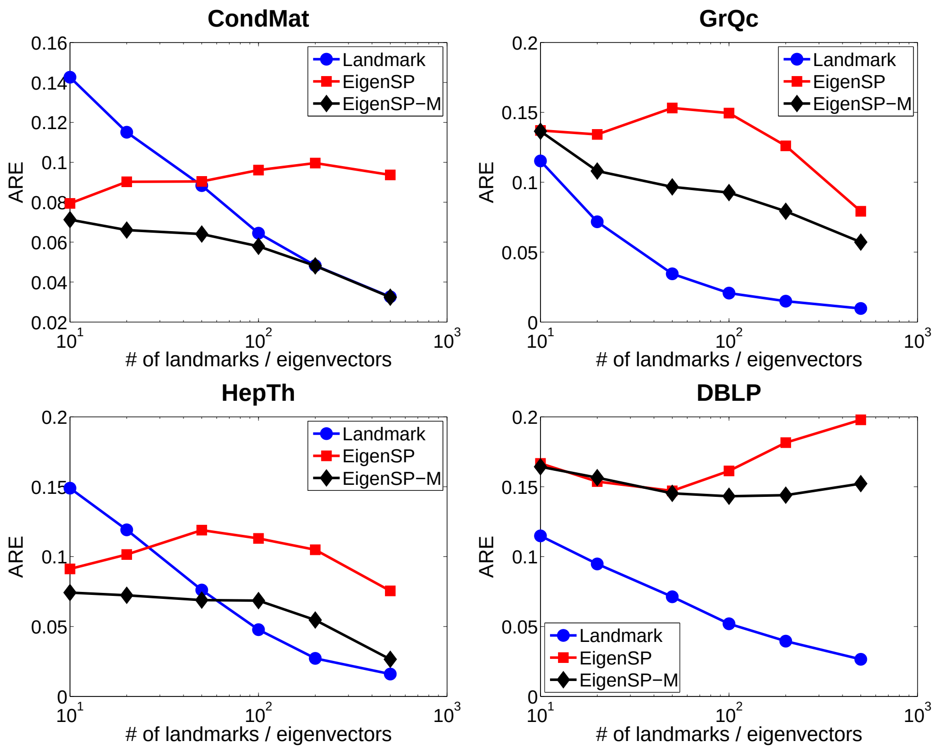


Figure 6.14: Average relative error (ARE) of datasets which have long average distances about 6 or 7 (Table 6.2) with many eigenvalues of landmarks. The accuracies of EigenSP-M are better than those of EigenSP if we use many eigenvalues on these datasets. Horizontal axis: the number of eigenvectors or landmarks. Vertical axis: ARE.

of 2. It takes about 1.2s on average for a graph with 55M edges (IMDb) when calculating distances of 2 (not shown in Figure 6.15). This should be because BFS, in real networks, will check many nodes connected to some *hub* nodes very often, even if the distance of the node pair is short. Our methods and the landmark methods can estimate far faster than BFS, i.e., within the order of tens of microseconds.

6.4.7 Precomputation Time

We evaluate the scalability of our methods from the precomputing point of view. The most expensive precomputation of our methods is the eigenvalue computation. We used the MATLAB command *eigs*, which calculates them

methods	BFS	EigenSP	triangulated	GNP
IMDb	1289.5 ms	13.1 μs	25.5 μs	3.5 μs
AS-CAIDA	2.4 ms	12.0 μs	23.3 μs	3.4 μs
Enron	8.2 ms	15.2 μs	23.2 μs	3.5 μs
AstroPh	6.8 ms	14.5 μs	23.3 μs	3.5 μs
CondMat	5.8 ms	18.4 μs	23.3 μs	3.4 μs
GrQc	1.5 ms	23.6 μs	22.9 μs	3.3 μs
HepTh	2.5 ms	22.2 μs	23.0 μs	3.3 μs
DBLP	238.3 ms	26.5 μs	23.0 μs	3.5 μs

Table 6.3: The average query time of each method. EigenSP, the triangulated heuristic and GNP can calculate estimations within the order of microseconds, much faster than BFS.

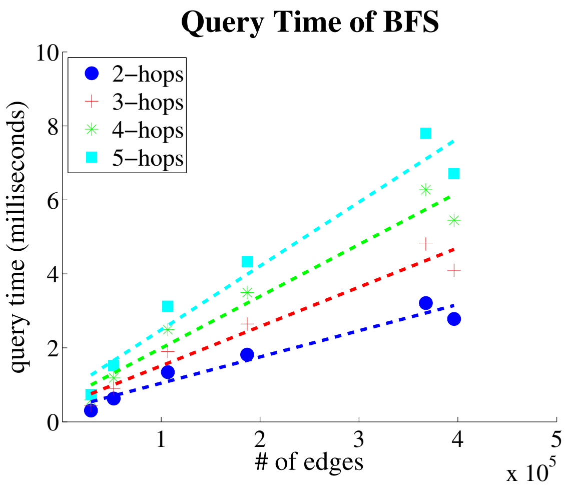


Figure 6.15: The numbers of edges of each dataset are plotted against the average query time of BFS. The query time of BFS increases almost linearly with the number of edges. It takes more than $1ms$ for graphs with more than $100K$ edges, even in calculating distances of 2.

based on the Lanczos algorithm [40]. We show the computation time of the top 10 eigenvalues in Table 6.4. The computation time is about 1 minute for the graph with 50M edges. Figure 6.16 is a log-log plot of the number of edges against the precomputation time of EigenSP. The slope in the log-log plot is 0.72, and it suggests that the precomputation time may increase at

methods	EigenSP	triangulated	GNP
IMDb	64.8 *s*	473.3 *s*	N.A.
AS-CAIDA	0.6 *s*	2.3 *s*	122.8 *min*
Enron	1.5 *s*	3.6 *s*	157.7 *min*
AstroPh	0.7 *s*	3.6 *s*	84.0 *min*
CondMat	0.8 *s*	2.4 *s*	103.5 *min*
GrQc	0.3 *s*	0.5 *s*	24.6 *min*
HepTh	1.0 *s*	0.9 *s*	44.2 *min*
DBLP	12.4 *s*	36.7 *s*	17.0 *hour*

Table 6.4: The precomputation time. Precomputing EigenSP and the triangulated heuristic is fast enough for all the datasets, whereas GNP's is inefficient.

most linearly with the number of edges, and that they may be computed within, at most, several hours for a graph with billions of edges.

We should mention the precomputation time of the landmark methods. Table 6.4 also shows that precomputation of the triangulated heuristic is fast enough for all of the datasets. Because the computation of the Simplex Downhill method is very expensive, the precomputation of GNP takes a very long time compared to the triangulated heuristic and EigenSP.

6.5 Conclusion

We devised a novel method, called EigenSP, that estimates shortest path distances by using an adjacency matrix approximated by a few eigenvalues and eigenvectors. We found that the values of the elements of eigenvectors vary widely about the *exact* values of the elements without depending on the number of eigenvectors, and we added a simple heuristic to EigenSP (resulting in EigenSP-M). The experimental results on eight large-scale real-world graphs show that the AREs of our methods are lower than those of the landmark-based method on datasets with short average distances of about 4, which are very close to that of Facebook [10]. Moreover, we showed that our methods can estimate shorter distances better than the landmark-based method on such datasets. A property like this is beneficial to applications such as social network analysis which try to identify *potential friends* separated by short distances. In addition, it should be possible to combine our method with the landmark-based method. The simplest

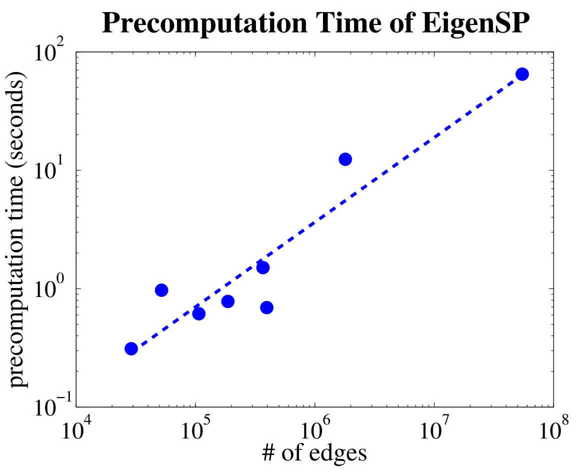


Figure 6.16: The numbers of edges of each dataset are plotted against the precomputation time of EigenSP in the log-log plot. The slope in the log-log plot is 0.72, and it suggests that the precomputing time may increase at most linearly with the number of edges.

way would be to use the landmark-based method for longer distance and trust our method for shorter distance, and we can consider many other combining strategies. Note that the accuracy of the landmark-based method on datasets with short average distances can be significantly improved by storing the actual path information [63], and applying such information to our methods is a future research direction.

Recently, Akiba *et al.* proposed a novel method for shortest-path distance queries, which precomputed *distance labels* for nodes by performing a breadth-first search from every nodes [6]. It can answer even *exact* distances within microseconds. It should be also a future direction to evaluate the advantage and disadvantage of our method compared to this method.

Chapter 7

Conclusion

Now we describe the concluding remarks. Contributions of this thesis is summarized in the first section, and we conclude with future research directions in the following section.

7.1 Summary of the Contributions

In this section we summarize our works and contributions described in this thesis.

Novel anomaly detection method of k,k-hypergraph. (Chapter 4) We defined the problem of detecting *anomalous* k,k-hypercliques, and proposed a novel method which automatically detects such anomalies using CP decomposition. The advantage of our method is that we extract anomalous communities in both views of the size of communities and the path capacities between the communities and the *important* nodes, by utilizing even very low absolute values of the eigenvectors of CP decomposition. We have shown that our method can detect synthesized anomalies effectively, and the computation time scale linearly at the size of hyperedges, leveraged by a sparse coding of CP decomposition. Also, we introduced some visualization methods that users can easily understand nature of detected anomalies, and some examples of detected anomalies were shown by empirical experiments using real-world datasets from three dissimilar application domains.

Pattern discovery of bipartite graph. (Chapter 5) We proposed some novel *patterns* of the nodes related to community structures in graph data, which we can observe by inspecting distributions of values of elements of singular vectors and weighted degrees corresponding to nodes. These *patterns* include a pattern of the ratio of the degrees and the numbers of paths to a community, a pattern of the ratio of connections straddling two communities, and so on. And we introduced a framework which can spot these *patterns* by detecting *linear-like patterns* in 2D-plots of node properties, using Hough transform. None of existing works on anomaly detection in graph data consider the path capacities between all nodes and communities, *i.e.*, determine *normal* connection patterns between nodes and communities. We showed our framework could spot some interesting *patterns* in real-world datasets with millions of edges.

Fast shortest path distance estimation method. (Chapter 6). We proposed a novel method to estimate shortest path distance of query node pairs within order of micro seconds, using eigen decomposition. We also showed the accuracy of approximation of values of an adjacency matrix varies according to the choice of number of eigenvalues, and it even became worse when we use many eigenvalues. We introduced a heuristic to tackle this problem. By conducting some experiments using eight real-world datasets, we showed our method could estimate shorter distances more accurately than existing well-known methods like the landmark-based methods.

7.2 Future Works

As future works, there is a big room to improve our methods introduced in this thesis.

Parameter settings of spike detection algorithm. For anomaly detection method of k,k-hypergraph proposed in Chapter 4, we should have a clear strategy to select parameters of spike detection algorithm, the heart of this method.

Anomaly detection based on *patterns* detected. As our goal is anomaly detection, we should introduce some methods to detect anomalies or outliers based on *patterns* detected by our method proposed in Chapter 5. In addition, there are possibly other *patterns* than *linear-like patterns*, and other technique will be required to detect such *patterns* in future.

Improvement of the distance estimation method and applications. Even though our method introduced in Chapter 6 is very simple, there are many possibilities to improve our method. One possible improvement is better selection strategy of eigenvalues and eigenvectors, as they are not necessarily selected from the largest eigenvalues. For example, we can estimate mean error and mean squared error of approximation of multiplied adjacency matrix as Figure 6.6 and Figure 6.7, and greedily pick up eigenvalues and eigenvectors from those reduce errors most efficiently by adding them.

Another possible improvement is the combination with the landmark-based methods. The simplest way would be to use the landmark-based method for longer distance and trust our method for shorter distance. Other way would be to divide whole graph into some partitions. We could estimate distances that straddle partitions by using the landmark-based methods, and estimate distances within partitions by using our method.

Moreover, there is some possibility of using our method to improve algorithm for point to point shortest path queries. ALT algorithms combine the bidirectional Dijkstra algorithm with the A* algorithm and landmarks in order to prune the search space [61, 79]. They use graph-theoretic lower-bounding technique based on landmarks and the triangle inequality. But these lower bounds might be too low to prune the search space if two nodes are located far from any landmark nodes, which should happen as the A* search approaches the query nodes. Even though our method cannot provide lower bounds, our method might be able to estimate short distance near the query nodes, and give good approximate shortest paths effectively.

Furthermore, there are many more complicated problems beyond the problems we tackled in this thesis. One of the problems is how to detect *novel anomalous patterns*. For example, a serious problem for analysts in charge of a company's security system is how to distinguish *remarkable* attacks which have never been seen before, from vast amount of *ordinary* attacks in the intrusion detection systems logs. In other words, this is a problem of distinguishing *anomalous* connection patterns from *normal* connection patterns as we spotted in Chapter 5, so far we distinguish them manually. This is a hard problem because it is hardly trivial to determine that a today's *pattern* is similar to some past *patterns* caused by similar kind of attacks. Another problem is how to deal with datasets containing information at various levels of granularity. For example, there are many purchasing data which include records with only the receipt numbers and product names, and records with the customer IDs in addition to the receipt numbers and product names, and records with the customer's zip codes, and many more. Obviously we cannot deal with such datasets by using a simple technique like the basic tensor decomposition techniques. However, it should be a natural situation that we have to analyze such datasets, in the era that more and more data can be generated and stocked from everywhere in the society.

Finally, anomaly detection of graph data is almost endless problem. This is very hard task, but practical solutions are desired intensely in the wide variety of applications.

Bibliography

[1] Berkeley Drosophilia Genome Project. Patterns of gene expression in Drosophila embryogenesis. `http://insitu.fruitfly.org/cgi-bin/ex/insitu.pl`. 40, 63

[2] Lawrence Berkeley National Laboratory and ICSI. LBNL/ICSI enterprise tracing project. `http://www.icir.org/enterprise-tracing/`. 49

[3] I. Abraham, Y. Bartal, H. T.-H. Chan, K. Dhamdhere, A. Gupta, J. M. Kleinberg, O. Neiman, and A. Slivkins. Metric embeddings with relaxed guarantees. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS '05)*, pages 83–100, 2005. 16

[4] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup. Scalable tensor factorizations with missing data. In *Proceedings of the 2010 SIAM International Conference on Data Mining (SDM '10)*, pages 701–712, 2010. 18

[5] C. C. Aggarwal and P. S. Yu. Outlier detection for high dimensional data. *SIGMOD Record*, 30:37–46, May 2001. 9

[6] T. Akiba, Y. Iwata, and Y. Yoshida. Fast exact shortest-path distance queries on large networks by pruned landmark labeling. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD '13)*, pages 349–360, 2013. 122

[7] L. Akoglu, M. McGlohon, and C. Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *proceedings of the 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD '10)*, pages 410–421, 2010. 3, 5, 11, 44, 70, 73, 90

[8] C. A. Andersson and R. Bro. The N-way toolbox for MATLAB. *Chemometrics and Intelligent Laboratory Systems*, 52(1):1–4, 2000. 18

[9] A. Arning, R. Agrawal, and P. Raghavan. A linear method for deviation detection in large databases. In *Proceedings of the second ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '96)*, pages 164–169, 1996. 9, 10

[10] L. Backstrom, P. Boldi, M. Rosa, J. Ugander, and S. Vigna. Four degrees of separation. In *Proceedings of the 3rd Annual ACM Web Science Conference (WebSci '12)*, pages 33–42, 2012. 121

[11] B. W. Bader, T. G. Kolda, et al. Matlab tensor toolbox version 2.5. Available online, January 2012. `http://www.sandia.gov/~tgkolda/TensorToolbox/`. 18, 58, 61

[12] D. A. Bader, S. Kintali, K. Madduri, and M. Mihail. Approximating betweenness centrality. In *Proceedings of the 5th international conference on Algorithms and models for the web-graph (WAW'07)*, volume 4863, pages 124–137, 2007. 15

[13] H. B. Barlow. Unsupervised learning. *Neural Computation*, 1:295–311, 1989. 9

[14] V. Barnett and T. Lewis. *Outliers in Statistical Data.* John Wiley and Sons, 1994. 8

[15] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL Web Ontology Language Reference. Technical report, W3C, http://www.w3.org/TR/owl-ref/, February 2004. 44

[16] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 5(3):1–22, 2009a. 44

[17] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. Dbpedia - a crystallization point for the web of data. *Journal of Web Semantics*, 7(3):154–165, 2009b. 44

[18] P. Bonacich. Factoring and weighting approaches to status scores and clique identification. *Journal of Mathematical Sociology*, 2(1):113–120, 1972. 15

[19] S. P. Borgatti. Centrality and network flow. *Social Networks*, 27(1):55–71, 2005. 2

[20] S. P. Borgatti and M. G. Everett. A graph-theoretic perspective on centrality. *Social Networks*, 28(4):466–484, 2006. 2, 15

[21] D. M. Boyd and N. Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1):210–230, 2007. 44

[22] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(163), 2001. 15

[23] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: identifying density-based local outliers. *SIGMOD Record*, 29:93–104, May 2000. 8, 9

[24] S. Brin and L. Page. The anatomy of a large-scale hypertextual (web) search engine. In *Proceedings of the seventh international conference on World Wide Web 7 (WWW7)*, pages 107–117, 1998. 15

[25] A. Bronstein, J. Das, M. Duro, R. Friedrich, G. Kleyner, M. Mueller, S. Singhal, and I. Cohen. Self-aware services: Using bayesian networks for detecting anomalies in internet-based services. In *Proceedings of the 7th IFIP/IEEE International Symposium on Integrated Network Management (IM 2001)*, pages 623–638, 2001. 9

[26] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011. 10

[27] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*, pages 1306–1313, 2010a. 39, 63

[28] A. Carlson, J. Betteridge, R. C. Wang, E. R. Hruschka, Jr., and T. M. Mitchell. Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining (WSDM '10)*, pages 101–110, 2010b. 39

[29] J. Carroll and J.-J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of 'eckart-young' decomposition. *Psychometrika*, 35(3):283–319, September 1970. 18

[30] D. Chakrabarti. Autopart: Parameter-free graph partitioning and outlier detection. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2004)*, pages 112–124, 2004. 3, 10, 44, 70

[31] D. Chakrabarti, S. Papadimitrou, D. Modha, and C. Faloutsos. Fully automatic cross-associations. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '04)*, pages 79–88, 2004. 13

[32] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41:15:1–15:58, July 2009. 8

[33] V. Chaoji, M. A. Hasan, S. Salem, and M. J. Zaki. SPARCL: Efficient and effective shape-based clustering. In *Proceedings of the 2008 IEEE International Conference on Data Mining (ICDM '08)*, pages 93–102, 2008. 9

[34] A. Chaudhary, A. S. Szalay, and A. W. Moore. Very fast outlier detection in large multidimensional data sets. In *Proceedings of the 7th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery (DMKD '02)*, 2002. 9

[35] E. C. Chi and T. G. Kolda. On tensors, sparsity, and nonnegative factorizations. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1272–1299, 2012. 9, 18

[36] F. R. K. Chung. *Spectral Graph Theory.* American Mathematical Society, Providence, RI, USA, 1997. 12, 14

[37] A. Clauset, M. E. J. Newman, and C. Moore. Finding Community Structure in Very Large Networks. *Physical Review*, 2004. 13

[38] D. J. Cook and L. B. Holder. Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research*, 1:231–255, February 1994. 44

[39] M. Costa, M. Castro, A. I. T. Rowstron, and P. B. Key. Pic: Practical internet coordinates for distance estimation. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, pages 178–187, 2004. 17

[40] J. K. Cullum and R. A. Willoughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Vol. 1.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002. 101, 120

[41] F. Dabek, R. Cox, M. F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '04)*, pages 15–26, 2004. 17

[42] K. Das and J. G. Schneider. Detecting anomalous records in categorical datasets. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '07)*, pages 220–229, 2007. 9

[43] K. Das, J. G. Schneider, and D. B. Neill. Anomaly pattern detection in categorical datasets. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '08)*, pages 169–176, 2008. 9

[44] L. Dehaspe and H. Toivonen. Discovery of frequent datalog patterns. *Data Mining and Knowledge Discovery*, 3:7–36, March 1999. 44

[45] M. J. Desforges, P. J. Jacob, and J. E. Cooper. Applications of probability density estimation to the detection of abnormal conditions in engineering. In *Proceedings of Institute of Mechanical Engineers*, volume 212, pages 687–703, 1998. 9

[46] I. Dhillon, Y. Guan, and B. Kullis. Weighted Graph Cuts without EigenVectors : A Multilevel Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1944–1957, November 2007. 12

[47] I. Dhillon, S. Mallela, and D. Modha. Information-theoretic co-clustering. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '03)*, pages 89–98, 2003. 13

[48] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959. 16, 98

[49] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972. 81

[50] W. Eberle and L. Holder. Discovering structural anomalies in graph-based data. In *Proceedings of the ICDM 2007 Workshop on Mining Graphs and Complex Structures (MGCS)*, pages 393–398, 2007. 3, 10, 44, 70

[51] E. Eskin. Anomaly detection over noisy data using learned probability distributions. In *Proceedings of the 17th international conference on Machine learning (ICML '00)*, pages 255–262, 2000. 9

[52] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the second ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '96)*, pages 226–231, 1996. 9

[53] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010. 11, 25

[54] T. Franz, A. Schultz, S. Sizov, and S. Staab. TripleRank: Ranking semantic web data by tensor decomposition. In *Proceedings of the 8th International Semantic Web Conference (ISWC '09)*, pages 213–228, 2009. 46

[55] L. Freeman. Centrality in Social Networks: Conceptual Clarification. *Social Networks*, 1:215–239, 1979. 15

[56] L. C. Freeman. The gatekeeper, pair-dependency and structural centrality. *Quality and Quantity*, V14(4):585–592, 1980. 15

[57] R. Fujimaki, T. Yairi, and K. Machida. An approach to spacecraft anomaly detection problem using kernel feature space. In *Proceedings of the 11th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '05)*, pages 401–410, 2005. 10

[58] J. Gao, F. Liang, W. Fan, C. Wang, Y. Sun, and J. Han. On community outliers and their efficient detection in information networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '10)*, pages 813–822, 2010. 3, 11

[59] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB '99)*, pages 518–529, 1999. 9

[60] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12):7821–7826, June 2002. 12

[61] A. V. Goldberg and C. Harrelson. Computing the shortest path: A^* search meets graph theory. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms (SODA '05)*, pages 156–165, 2005. 16, 125

[62] J. Gómez, C. Gil, N. Padilla, R. Baños, and C. Jiménez. Design of a snort-based hybrid intrusion detection system. In *Proceedings of the 10th International Work-Conference on Artificial Neural Networks (IWANN '09)*, pages 515–522, 2009. 44

[63] A. Gubichev, S. Bedathur, S. Seufert, and G. Weikum. Fast and accurate estimation of shortest paths in large graphs. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM '10)*, pages 499–508. ACM, 2010. 122

[64] S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5):345–366, 2000. 9

[65] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. *Information Systems*, 26(1):35–58, 2001. 9

[66] M. Gupta, J. Gao, and J. Han. Community distribution outlier detection in heterogeneous information networks. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2013)*, pages 557–573, 2013. 3, 11

[67] B. H. Hall, A. B. Jaffe, and M. Trajtenberg. The NBER patent citation data file: Lessons, insights and methodological tools. Working Paper 8498, National Bureau of Economic Research, October 2001. 40, 84

[68] F. Harary and A. J. Schwenk. The spectral approach to determining the number of walks in a graph. *Pacific Journal of Mathematics*, 80(2):443–449, 1979. 3, 14

[69] R. Harshman. Foundations of the parafac procedure: Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16, 1970. 18

[70] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, SSC-4(2):100–107, 1968. 16

[71] J. Håstad. Tensor rank is NP-complete. *Journal of Algorithms*, 11:644–654, December 1990. 50

[72] D. Hawkins. *Identification of outliers (Monographs on Statistics & Applied Probability)*. Chapman and Hall, 1980. 8

[73] K. Hayashi, T. Takenouchi, T. Shibata, Y. Kamiya, D. Kato, K. Kunieda, K. Yamada, and K. Ikeda. Exponential family tensor factorization for missing-values prediction and anomaly detection. In *Proceedings of the 2010 IEEE International Conference on Data Mining (ICDM '10)*, pages 216–225, 2010. 9, 18

[74] Z. He, X. Xu, and S. Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10):1641–1650, 2003. 9

[75] A. Hlaoui and S. Wang. A direct approach to graph clustering. In *Proceedings of the IASTED International Conference on Neural Networks and Computational Intelligence (NCI 2004)*, pages 158–163, 2004. 12

[76] L. B. Holder, D. J. Cook, and S. Djoko. Substucture discovery in the subdue system. In *Proceedings of the Workshop on Knowledge Discovery in Databases (KDD Workshop)*, pages 169–180, 1994. 10

[77] P. Holme, M. Huss, and H. Jeong. Subnetwork hierarchies of biochemical pathways. *Bioinformatics*, 19(4):532–538, 2003. 12

[78] T. Idé and H. Kashima. Eigenspace-based anomaly detection in computer systems. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '04)*, pages 440–449, 2004. 11

[79] T. Ikeda, M.-Y. Hsu, H. Imai, S. Nishimura, H. Shimoura, T. Hashimoto, K. Tenmoku, and K. Mitoh. A fast algorithm for finding better routes by AI search techniques. In *Proceedings of the Vehicle Navigation and Information Systems (VNIS '94)*, pages 291–296, 1994. 16, 125

[80] M. Ji, J. Han, and M. Danilevsky. Ranking-based classification of heterogeneous information networks. In *Proceeding of the 17th ACM*

SIGKDD international conference on Knowledge discovery and data mining (KDD '11), pages 1298–1306, 2011. 45

[81] W. Jin, A. K. H. Tung, and J. Han. Mining top-n local outliers in large databases. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '01)*, pages 293–298, 2001. 9

[82] R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall, 1998. 8

[83] U. Kang and C. Faloutsos. Beyond 'caveman communities': Hubs and spokes for graph compression and mining. In *Proceedings of the 2011 IEEE International Conference on Data Mining (ICDM '11)*, pages 300–309, 2011. 31

[84] U. Kang, B. Meeder, and C. Faloutsos. Spectral analysis for billion-scale graphs: Discoveries and implementation. In *proceedings of the 15th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD '11)*, pages 13–25, 2011. 14

[85] U. Kang, E. E. Papalexakis, A. Harpale, and C. Faloutsos. Gigatensor: scaling tensor analysis up by 100 times - algorithms and discoveries. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '12)*, pages 316–324, 2012. 18

[86] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1999. 12

[87] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, March 1953. 15

[88] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley, 1990. 9

[89] E. J. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards parameter-free data mining. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '04)*, pages 206–215, 2004. 10

[90] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1998. 4, 15

[91] J. M. Kleinberg, A. Slivkins, and T. Wexler. Triangulation and embedding using small sets of beacons. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS '04)*, pages 444–453, 2004. 16, 98

[92] G. Klyne and J. J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. World Wide Web Consortium, Recommendation REC-rdf-concepts-20040210, February 2004. 44

[93] E. Knorr and R. Ng. Finding intentional knowledge of distance-based outliers. In *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB '99)*, pages 211–222, 1999. 9

[94] E. M. Knorr and R. T. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB '98)*, pages 392–403, 1998. 8, 9

[95] E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based outliers: algorithms and applications. *The International Journal on Very Large Data Bases (The VLDB Journal)*, 8(3-4):237–253, 2000. 9

[96] T. Kohonen. *Self-organizing maps.* Springer series in information sciences, 30. Springer, Berlin, 3rd edition, December 2001. 9

[97] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, September 2009. 5, 18, 22, 28, 46

[98] T. G. Kolda and J. Sun. Scalable tensor decompositions for multi-aspect data mining. In *Proceedings of the 2008 IEEE International Conference on Data Mining (ICDM '08)*, pages 363–372, 2008. 18

[99] M. Kuramochi and G. Karypis. An efficient algorithm for discovering frequent subgraphs. *IEEE Transactions on Knowledge and Data Engineering*, 16:1038–1051, September 2004. 44

[100] P. A. Lawrence. *The Making of a Fly: The Genetics of Animal Design.* Wiley-Blackwell, 1992. 66

[101] A. Lazarevic and V. Kumar. Feature bagging for outlier detection. In *Proceedings of the 11th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '05)*, pages 157–166, 2005. 9

[102] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems: Proceedings of the 2000 Conference (NIPS)*, pages 556–562, 2000. 18

[103] Y.-R. Lin, J. Sun, P. Castro, R. B. Konuru, H. Sundaram, and A. Kelliher. Metafac: community discovery via relational hypergraph factorization. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '09)*, pages 527–536, 2009. 13

[104] X. Liu and T. Murata. Detecting communities in k-partite k-uniform (hyper)networks. *Journal of Computer Science and Technology*, 26(5):778–791, 2011. 13

[105] L. Lu and T. Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011. 2

[106] A. S. Maiya and T. Y. Berger-Wolf. Benefits of bias: Towards better characterization of network sampling. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '11)*, pages 105–113, 2011. 98

[107] K. Makino and T. Uno. New algorithms for enumerating all maximal cliques. In *Proceedings of the 9th Scandinavian Workshop on Algorithm Theory (SWAT '04)*, pages 260–272, 2004. 14

[108] Y. Mao, L. K. Saul, and J. M. Smith. Ides: An internet distance estimation service for large networks. *IEEE Journal on Selected Areas in Communications*, 24(12):2273–2284, 2006. 16, 98

[109] K. Maruhashi and C. Faloutsos. EigenDiagnostics: Spotting connection patterns and outliers in large graphs. In *Proceedings of the ICDM 2010 Workshop on Large-scale Analytics for Complex Instrumented Systems (LACIS 2010)*, pages 1328–1337, 2010. 6, 70

[110] K. Maruhashi, F. Guo, and C. Faloutsos. MultiAspectForensics: Pattern mining on large-scale heterogeneous networks with tensor analysis. In *Proceedings of the 2011 International Conference on Advances in Social Networks Analysis and Mining (ASONAM '11)*, pages 203–210, 2011. 5, 44

[111] K. Maruhashi, F. Guo, and C. Faloutsos. MultiAspectForensics: mining large heterogeneous networks using tensor. *International Journal of Web Engineering and Technology (IJWET)*, 7(4):302–322, 2012. 5, 44

[112] K. Maruhashi, J. Shigezumi, N. Yugami, and C. Faloutsos. EigenSP: A more accurate shortest path distance estimation on large-scale networks. In *Proceedings of the ICDM 2012 Workshop on Data Mining in Networks (DaMNet 2012)*, pages 234–241, 2012. 6, 97

[113] K. Maruhashi and N. Yugami. MultiAspectSpotting: Spotting anomalous behavior within count data using tensor. In *Proceedings of the 18th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD '14)*, 2014. To appear. 10

[114] P. Miettinen. Boolean tensor factorizations. In *Proceedings of the 2011 IEEE International Conference on Data Mining (ICDM '11)*, pages 447–456, 2011. 19, 59

[115] P. Miettinen, T. Mielikäinen, A. Gionis, G. Das, and H. Mannila. The discrete basis problem. *IEEE Transactions on Knowledge and Data Engineering*, 20(10):1348–1362, 2008. 19

[116] B. Mohar. The Laplacian spectrum of graphs. *Graph Theory, Combinatorics, and Applications*, 2:871–898, 1991. 12

[117] B. Mohar and M. Juvan. Some applications of laplace eigenvalues of graphs. In *Graph Symmetry: Algebraic Methods and Applications, volume 497 of NATO ASI Series C*, volume 497, pages 227–275, 1997. 12

[118] A. L. Montgomery, S. Li, K. Srinivasan, and J. C. Liechty. Modeling online browsing and path analysis using clickstream data. *Marketing Science*, 23(4):579–595, 2004. 40, 85

[119] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965. 16, 113

[120] N. Neubauer and K. Obermayer. Towards Community Detection in k-Partite k-Uniform Hypergraphs. In *Workshop on Analyzing Networks and Learning with Graphs, NIPS 2009.* 13

[121] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(036104), 2006. 2, 3, 13

[122] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 103:8577–8582, 2006. 2, 3, 13

[123] A. Ng, M. Jordan, and Y. Weiss. On Spectral Clustering : Analysis and an Algorithm. *Advances in Neural Information Processing Systems: Proceedings of the 2002 Conference (NIPS)*, 2002. 12

[124] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB '94)*, pages 144–155, 1994. 9

[125] T. S. E. Ng and H. Zhang. Predicting internet network distance with coordinates-based approaches. In *Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, 2002. 6, 16, 98, 112, 113

[126] C. C. Noble and D. J. Cook. Graph-based anomaly detection. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '03)*, pages 631–636, 2003. 3, 10, 44, 70

[127] J. D. Noh and H. Rieger. Random walks on complex networks. *Physical Review Letters*, 92:118701, 2004. 15

[128] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, June 2005. 14

[129] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney. A first look at modern enterprise traffic. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement (IMC '05)*, pages 2–2, 2005. 38, 49, 61, 84

[130] R. Pang, M. Allman, V. Paxson, and J. Lee. The devil and packet trace anonymization. *SIGCOMM Computer Communication Review*, 36:29–38, January 2006. 39

[131] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos. Loci: Fast outlier detection using the local correlation integral. In *Proceedings of the 19th International Conference on Data Engineering (ICDE '03)*, pages 315–, 2003. 9

[132] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos. Parcube: Sparse parallelizable tensor decompositions. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2012)*, pages 521–536, 2012. 18

[133] J. Pei, D. Jiang, and A. Zhang. On mining cross-graph quasi-cliques. In *Proceedings of the 11th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '05)*, 2005. 14

[134] M. Potamias, F. Bonchi, C. Castillo, and A. Gionis. Fast shortest path distance estimation in large networks. In *Proceedings of the 18th ACM International Conference on Information and Knowledge Management (CIKM '09)*, pages 867–876, 2009. 6, 16, 98, 112, 113

[135] B. A. Prakash, M. Seshadri, A. Sridharan, S. Machiraju, and C. Faloutsos. Eigenspokes: Surprising patterns and scalable community chipping in large graphs. In *proceedings of the 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD '10)*, pages 435–448, 2010. 14, 75, 76, 79

[136] M. J. Rattigan, M. E. Maier, and D. Jensen. Using structure indices for efficient approximation of network properties. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '06)*, pages 357–366, 2006. 98

[137] M. J. Rattigan, M. E. Maier, and D. Jensen. Graph clustering with network structure indices. In *Proceedings of the 24th international conference on Machine learning (ICML '07)*, pages 783–790, 2007. 12

[138] M. Sales-Pardo, R. Guimerà, A. A. Moreira, and L. a. N. Amaral. Extracting the hierarchical organization of complex systems. *Proceedings of the National Academy of Sciences of the United States of America*, 104(39):15224–9, Sept. 2007. 12

[139] S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007. 11, 25

[140] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001. 8

[141] A. Shashua and T. Hazan. Non-negative tensor factorization with applications to statistics and computer vision. In *Proceedings of the 22nd international conference on Machine learning (ICML '05)*, pages 792–799, 2005. 18

[142] J. Shi and J. Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 888–905, August 2000. 12

[143] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang. A novel anomaly detection scheme based on principal component classifier. In *Proceedings of the ICDM 2003 Workshop on Foundations and New Directions in Data Mining*, pages 171–179, 2003. 10

[144] A. Siebes, J. Vreeken, and M. van Leeuwen. Item sets that compress. In *Proceedings of the 2006 SIAM International Conference on Data Mining (SDM '06)*, pages 395–406, 2006. 10

[145] K. Smets and J. Vreeken. The odd one out: Identifying and characterising anomalies. In *Proceedings of the 2011 SIAM International Conference on Data Mining (SDM '11)*, pages 804–815, 2011. 10

[146] R. Smith, A. Bivens, M. Embrechts, C. Palagiri, and B. Szymanski. Clustering approaches for anomaly based intrusion detection. *Proceedings of intelligent engineering systems through artificial neural networks*, 2002. 9

[147] K. Stephenson and M. Zelen. Rethinking centrality: Methods and examples. *Social Networks*, 11(1):1–37, Mar. 1989. 15

[148] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos. Neighborhood formation and anomaly detection in bipartite graphs. In *Proceedings of the 2005 IEEE International Conference on Data Mining (ICDM '05)*, pages 418–425, 2005. 11, 44

[149] J. Sun, D. Tao, S. Papadimitriou, P. S. Yu, and C. Faloutsos. Incremental tensor analysis: Theory and applications. *ACM Transactions on Knowledge Discovery from Data*, 2:11:1–11:37, October 2008. 18

[150] J. Sun, Y. Xie, H. Zhang, and C. Faloutsos. Less is more: Compact matrix decomposition for large sparse graphs. *Proceedings of the 2007 SIAM International Conference on Data Mining (SDM '07)*, pages 366–377, 2007. 3, 10, 70

[151] Y. Sun, Y. Yu, and J. Han. Ranking-based clustering of heterogeneous information networks with star network schema. In *Proceeding of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '09)*, pages 797–806, 2009. 11, 45

[152] L. Tang and M. Crovella. Virtual landmarks for the internet. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement (IMC '03)*, pages 143–152, 2003. 16, 98

[153] P. Tomancak, A. Beaton, R. Weiszmann, E. Kwan, S. Shu, S. Lewis, S. Richards, M. Ashburner, V. Hartenstein, S. Celniker, and G. Rubin. Systematic determination of patterns of gene expression during Drosophila embryogenesis. *Genome Biology*, 3(12):research0088.1–0088.14, 2002. 40, 63

[154] P. Tomancak, B. Berman, A. Beaton, R. Weiszmann, E. Kwan, V. Hartenstein, S. Celniker, and G. Rubin. Global analysis of patterns of gene expression during Drosophila embryogenesis. *Genome Biology*, 8(7):R145, 2007. 40, 63

[155] G. Tomasi and R. Bro. A comparison of algorithms for fitting the parafac model. *Computational Statistics & Data Analysis*, 50(7):1700–1734, April 2006. 18

[156] H. Tong and C.-Y. Lin. Non-negative residual matrix factorization with application to graph anomaly detection. In *Proceedings of the 2011 SIAM International Conference on Data Mining (SDM '11)*, pages 143–153, 2011. 3, 11, 70

[157] H. Tong, S. Papadimitriou, J. Sun, P. S. Yu, and C. Faloutsos. Colibri: fast mining of large static and dynamic graphs. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '08)*, pages 686–694, 2008. 3, 10, 44, 70

[158] H. Tong, B. A. Prakash, C. E. Tsourakakis, T. Eliassi-Rad, C. Faloutsos, and D. H. Chau. On the vulnerability of large graphs. In *Proceedings of the 2010 IEEE International Conference on Data Mining (ICDM '10)*, pages 1091–1096, 2010. 14

[159] C. E. Tsourakakis. MACH: Fast randomized tensor decompositions. In *Proceedings of the 2010 SIAM International Conference on Data Mining (SDM '10)*, pages 689–700, 2010. 18

[160] C. E. Tsourakakis. Counting triangles in real-world networks using projections. *Knowledge and Information Systems (KAIS)*, 26(3):501–520, 2011. 17

[161] A. Ukkonen, C. Castillo, D. Donato, and A. Gionis. Searching the wikipedia with contextual information. In *Proceedings of the 17th ACM International Conference on Information and Knowledge Management (CIKM '08)*, pages 1351–1352, 2008. 98

[162] T. Uno. An efficient algorithm for solving pseudo clique enumeration problem. *Algorithmica*, 56(1):3–16, 2010. 14

[163] M. V. Vieira, B. M. Fonseca, R. Damazio, P. B. Golgher, D. de Castro Reis, and B. A. Ribeiro-Neto. Efficient search ranking in social networks. In *Proceedings of the 16th ACM International Conference on Information and Knowledge Management (CIKM '07)*, pages 563–572, 2007. 98

[164] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007. 3, 12

[165] Y. Wang, S. Parthasarathy, and S. Tatikonda. Locality sensitive outlier detection: A ranking driven approach. In *Proceedings of the 27th International Conference on Data Engineering (ICDE '11)*, pages 410–421, 2011. 9

[166] S. White and P. Smyth. A spectral clustering approach to finding communities in graphs. In *Proceedings of the 2005 SIAM International Conference on Data Mining (SDM '05)*, pages 76–84, 2005. 13

[167] W.-K. Wong, A. W. Moore, G. F. Cooper, and M. M. Wagner. Rule-based anomaly pattern detection for detecting disease outbreaks. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI-02)*, pages 217–223, 2002. 9

[168] W.-K. Wong, A. W. Moore, G. F. Cooper, and M. M. Wagner. Bayesian network anomaly pattern detection for disease outbreaks. In *Proceedings of the 20th international conference on Machine learning (ICML '03)*, pages 808–815, 2003. 9

[169] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM '02)*, pages 721–724, 2002. 44

[170] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of data (SIGMOD '96)*, pages 103–114, 1996. 9

[171] X. Zhao, A. Sala, C. Wilson, H. Zheng, and B. Y. Zhao. Orion: Shortest path estimation for large social graphs. In *Proceedings of the Third USENIX Workshop on Online Social Networks (WOSN)*, pages 9–9, 2010. 6, 16, 98, 113

[172] X. Zhao, A. Sala, H. Zheng, and B. Y. Zhao. Fast and scalable analysis of massive social graphs. *Arxiv preprint*, arXiv:1107.5114, 2011. 16, 98

[173] N. Zheng, Q. Li, S. Liao, and L. Zhang. Flickr group recommendation based on tensor decomposition. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval (SIGIR '10)*, pages 737–738, 2010. 46